MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

Lumina Desktop Customization Frameworks

OpenBSD in 2016;
The Year of the Secure Desktop

FreeBSD UEFI root on ZFS
and Windows dual boot

Using SSD as the level two cache
for the FreeBSD dual-controller storage array

FreeNAS Getting Started Guide:
Part 4, Plugins

MINIX - a class-based Operating System
Part II

INTERVIEW WITH
ALEXANDER TODOROV, FOUNDER OF MR. SENKO
FRANCK PORCHER, FOUNDER OF FRANCKY'S COMPUTER ENGINEERING

VOL 10 NO 09
ISSUE 08/2016 (85)
1898-9144

Simplify your Data Center

Meet iXRack — a customized and repeatable rack-scale infrastructure ideal for web-scale, virtualization, big data, private cloud, and virtually any enterprise business application.

- **Converged & Repeatable**
- **Customizable for VMware, Big Data, Cloud & Hyperscale**
- **Up to 70% Lower TCO Than AWS**
- **Scalable Deployment**

For more information on iXRack, visit **iXsystems.com/iXRack** today.

MAGAZINE BSD

EDITORS' WORD

Dear Readers,

Huge apology for the delay. We hope that the articles in this issue will compensate for the waiting time.

We are heading towards Autumn and through most of the BSD convention season. Did you participate? What did you like the most?

We also have a couple of new updates and changes in our BSD World Monthly News section, like PC-BSD changing its name to TrueOS. We will start off this issue with the recent release of Lumina version 1.0.0 and Ken Moore's "Lumina Desktop Customization Frameworks" article about it.

If you liked our previous issue and articles about MINIX, we have something for you. Rafael Santiago de Souza Netto shared his second article on the subject, "Minix - A Class-Based Operating System."

Next, you will find the last article in a series by Mikhail E. Zakharov, entitled "Using SSD as the Level Two Cache for the FreeBSD Dual-Controller Storage Array." Staying in FreeBSD Corner, you will read "FreeBSD UEFI Root on ZFS and Windows Dual Boot" by Kevin Bowling.

We are very happy to share an article on OpenBSD. We would love to have more OpenBSD covered, so if you are an expert in it and would like to share your knowledge, just let us know. In "OpenBSD in 2016: The Year of the Secure Desktop" by David Rodriguez you will read about the OpenBSD desktop in general as well as how to encrypt your disk.

Mark VonFange shared with us the 4th part of his "FreeNAS Getting Started Guide: Plugins".

This time we have 2 interviews for you, The first one with Alexander Todorov, Founder of Mr. Senko, and the second one with Franck Porcher, Founder of Francky's Computer Engineering.

In the end of this issue you will find, as always, Rob's Column, a big dose of economics, politics and security.

Marta & BSD Team

MAGAZINE BSD

Editor in Chief:

Marta Ziemianowicz

marta.ziemianowicz@software.com.pl

Contributing:

Ken Moore , Rafael Santiago de Souza Netto, Mikhail E. Zakharov, Kevin Bowling, Mark VonFange, David Rodriguez, Alexander Todorov, Franck Porche and Rob Somerville.

Top Betatesters & Proofreaders:

Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe, Mani Kanth and Mark VonFange.

Special Thanks:

Annie Zhang

Denise Ebery

DTP:

Marta Ziemianowicz

Senior Consultant/Publisher:

Paweł Marciniak

pawel@software.com.pl

CEO:

Joanna Kretowicz

joanna.kretowicz@software.com.pl

Publisher:

Hakin9 Media SK 02-676 Warsaw, Poland Postepu 17D Poland worldwide publishing editors@bsdmag.org www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

CONTENTS

CONTENTS

Interview

The UK Sunday Times reports that the Reuben Brothers are in advanced talks to sell a 50% stake of Global Switch in a £5bn deal to an Asian consortium, with a potential option to sell the remainder at a later date. With the increased interest China is showing in Western infrastructure, is the cloud becoming an even greater security risk to national security?

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BSDP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BSDP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost $75 USD. Computer based BSDA exams cost $150 USD. The price of the BSDP exams are yet to be determined.

Payments are made through our registration website:
https://register.bsdcertification.org//register/payment

WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
http://www.bsdcertification.org

Registration for upcoming exam events is available at our registration website:
https://register.bsdcertification.org//register/get-a-bsdcg-id

PC-BSD-Renamed TrueOS To Use LibreSSL, Linux DRM 4.7 Compatibility

In case you missed it last month, PC-BSD is completely re-branding itself as TrueOS, that's across the board for their desktop, server, and embedded editions while they will abandon the PC-BSD name. More details are coming to light on the inaugural TrueOS release.

Rolling-releases of TrueOS have been available in beta form while the first official milestone is approaching. TrueOS will continue to be based on FreeBSD just as PC-BSD was all along. TrueOS is currently tracking the soon-to-be-released FreeBSD 11.0, but with some added changes.

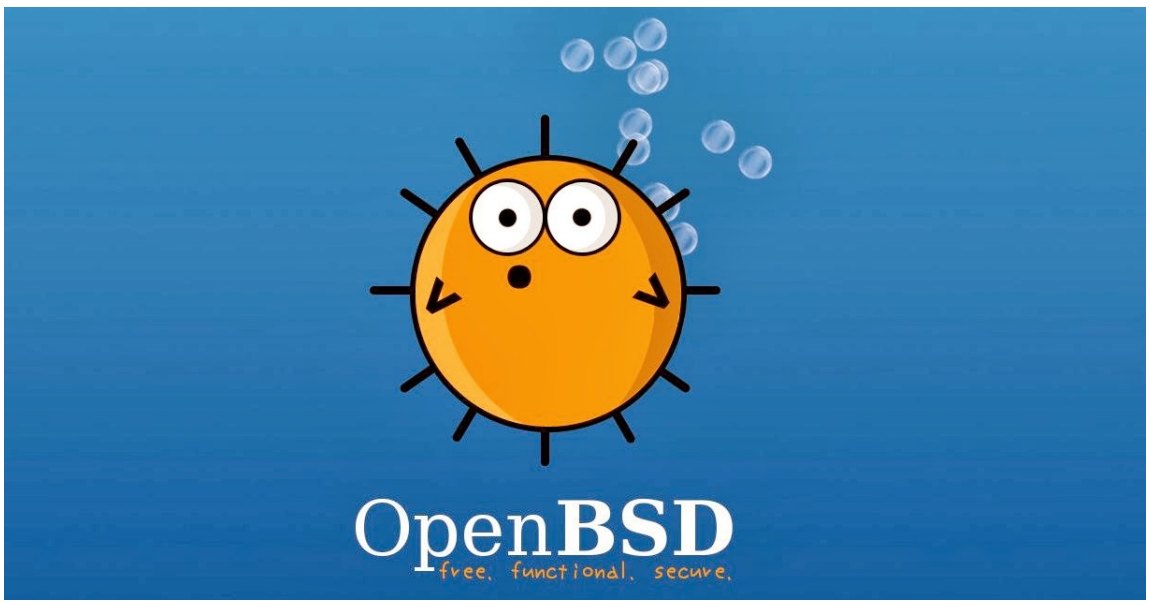
The differences with FreeBSD 11.0 being advertised by TrueOS includes replacing OpenSSL with LibreSSL, making use of the Linux 4.7 kernel DRM compatibility layer for better Intel graphics support, and using pkg to manage the entire system -- including FreeBSD base.

TrueOS will be following a rolling-release model that tracks FreeBSD -CURRENT, the TrueOS desktop will be built around the project's Qt5-powered Lumina Desktop, etc.

Those wishing to learn more about the latest details on TrueOS can read today's evolution blog post.

http://www.phoronix.com/scan.php?page=news_item&px=TrueOS-LibreSSL-DRM-4.7

OpenBSD 2016 Fundraising Campaign



The OpenBSD Foundation needs your help to achieve our fundraising goal of $250,000 for 2016.

Reaching this goal will ensure the continued health of the projects we support, will enable us to help them do more, and will avoid the distraction of financial emergencies that could spell the end of the projects.

2015 was a good year for the foundation financially, with one platinum, one gold, four silver and three bronze donors providing half of our total donations. Smaller contributions from 680 individuals provided the other half. While the total was down significantly after 2014's blockbuster year, we again exceeded our goal.

Our goal for 2016 is to increase the amount of support we offer for development, without compromising our regular support for the projects. We would like to:

- Plan and support more developer events (hackathons), and allow for more developers to attend these events.
- Continue to improve the project infrastructure.
- Fund more dedicated developer time for targeted development of specific projects.

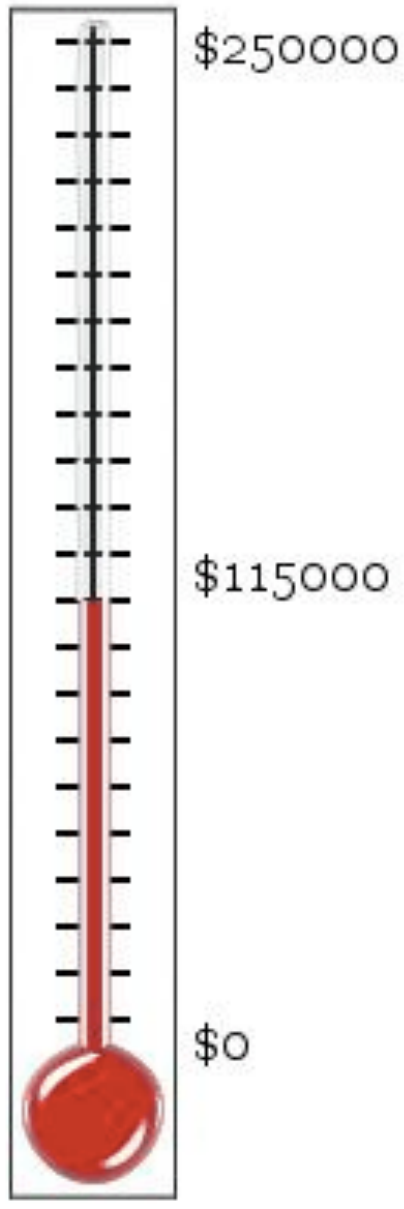
To achieve our goal we need both Individual and corporate sponsorship of the foundation. To put our goal in context:

- If $10 were given for every installation of OpenBSD in the last year from the master site (ignoring the mirrors) we would be at our goal.
- If $2 were given for every download of the OpenSSH source code in the last year from the master site (ignoring the mirrors) we would be at our goal.
- If a penny was donated for every pf or OpenSSH installed with a mainstream operating system or phone in the last year we would be at our goal.

As an individual or corporation, the best kind of donation we can receive is a recurring donation. This allows longer term planning on our part, instead of hoping for one time cash infusions. The easiest way for an individual to support us in this way is a recurring PayPal donation, which is our preference.

Donations to the foundation can be made on our Donations Page. We can be contacted regarding corporate sponsorship at *fundraising@openbsdfoundation.org*

Donation status on September 5th 2016:



https://bsdmag.org/openbsd_dontaions/

OpenBSD 6.0 Is Out With Better ARM Support, More SMP Fun, Dropped Linux Emulation

Kicking off September, the OpenBSD developers announced the release of OpenBSD 6.0.

Highlights for OpenBSD 6.0 include Linux-only binary emulation being removed due to being unmaintained and seldom used, updates to all the Open*/Libre packages, like LibreSSL and OpenSSH, continued work on SMP improvements, ARMv7 platform improvements, and W^X support being enabled by default for the base system.

Some of the ARMv7 work includes EFI bootloader support, a single kernel/ramdisk to support all SoCs, and more. OpenBSD 6.0 has also dropped the VAX architecture support.

Find out more about the many OpenBSD 6.0 improvements via OpenBSD.org.

http://www.phoronix.com/scan.php?page=news_item&px=OpenBSD-6.0-Released

Open Source Office Suite Apache OpenOffice Could Shut Down Soon

Apache OpenOffice, the open source office suite, is considering retirement. An email by the VP of OpenOffice has thrown light on the possible retirement of the office suite in the coming future. The project has not been able to push regular updates due to lack of volunteer developers, who have shifted to LibreOffice.

One of the major OpenOffice.org derivatives Apache OpenOffice may be packing its bags soon. The news is floating around that the open source office suite backed by the Apache Software Foundation won't see the light of the day in the coming future. An email sent by the Apache OpenOffice VP Dennis Hamilton is serving as the wellspring of speculations regarding the retirement of the OpenOffice.

"I have regularly observed that the Apache OpenOffice project has limited capacity for sustaining the project in an energetic manner. It is also my considered opinion that there is no ready supply of developers who have the capacity, capability, and will to supplement the roughly half-dozen volunteers holding the project together. It doesn't matter what the reasons for that might be."

— Hamilton wrote in the email.

Hamilton has listed a roadmap of how the retirement process will continue. The source code has been put on The Attic and the external libraries are not a part of the source code. The list also includes termination of the Apache OpenOffice blogs, social media accounts, and any future official release claims won't be related to the Apache OpenOffice.

Don't worry, LibreOffice is here to stay

OpenOffice has been on a downward popularity track ever since another OpenOffice.org spin-off LibreOffice is being considered as an alternative to Microsoft Office.

LibreOffice is being widely adopted, being a pre-installed office suite on many Linux distros. This has changed the minds of the contributing developers and many of them have started investing their efforts on LibreOffice. The lack of contributors has disabled Apache OpenOffice project's ability to push regular updates and bug fixes.

http://fossbytes.com/openoffice-may-close-its-doors-soon-libreoffice-is-a-great-alternative/

Open Source Leader Red Hat Lands On Forbes' "World's Most Innovative Companies" List

Red Hat Inc. has landed on the 25th position on Forbes' World's Most Innovative Companies list. Founded more than 20 years ago, Red Hat is the only open source company on the list. Red Hat CEO Jim Whitehurst has called this recognition a tribute to Red Hat associates around the world and the open source community.

Red Hat Inc., world's leading provider of Linux-based open source solutions, has announced that it has landed on Forbes' World's Most Innovative Companies list.

Red Hat on Forbes Lists

#25 Innovative Companies

#1,966 Global 2000

#820 in Market value

#25 Red Hat

Market Cap As of May 2016

$13.7 Billion

Ticker	RHT $73.67 ↓$-0.73 (-0.01%)
Industry	Software & Programming
Country	United States
President & Chief Executive Officer	James Whitehurst
Website	www.redhat.com
Employees	8,800
Sales	$2.05 B
Headquarters	Raleigh, North Carolina

The company is listed as the 25th most innovative company in the world. This recognition marks Red Hat's fourth appearance on this list (2012, 2014, 2015, 2015). Back in 2011, the company also found a place on Forbes' World's Most Innovative Growth Companies, according to Red Hat.

Red Hat was founded in 1993 with an aim of providing open-source software products for the enterprise. Over the years, in the open source community, Red Hat has become associated to enterprise operating system Red Hat Enterprise Linux, commonly known as RHEL.

Red Hat provides storage, middleware, applications, management products, support, and more. Red Hat is also known to contribute many free software projects. As of the current data, Red Hat is the second largest corporate contributor to Linux kernel.

Interestingly, Red Hat is the only open source software company on Forbes' World's Most Innovative Companies list.

Here's what Red Hat CEO and president Jim Whitehurst has to say about this development:

"It is always an honor to see Red Hat's name on this list. Today, open source is a default choice for innovation, and more than 90 percent of Fortune 500 companies rely on Red Hat's

solutions. This recognition by Forbes is a tribute to not only Red Hat associates around the world, but also to the open source communities that are driving community-powered innovation."

http://fossbytes.com/open-source-leader-red-hat-lands-on-forbes-worlds-most-innovative-companies-list/

Latest Intel, AMD chips will only run Windows 10 ... and Linux, BSD, OS X

El Reg answers your questions while you wait for make all to finish.

Water cooler I read an article this week headlined: "The latest Kaby Lake, Zen chips will support only Windows 10." It claimed Intel and AMD's new processors are "officially supported only by Microsoft's Windows 10." This can't be true? What about Linux?

Journalists, right? The short answer is Intel's Kaby Lake, aka its seventh-generation Core i3, i5 and i7 processors, and AMD's Zen-based chips, are not locked down to Windows 10, they'll boot Linux, the BSDs, Chrome OS, home-brew kernels, OS X, whatever software supports them.

So if you want to use Linux or some other non-Windows OS on your new CPUs, you'll be fine. It's OK, we checked.

Sweet. What about Windows 7? Or Windows 8.1? Or any Windows pre-10?

Yeah... nah. Shad Larsen, Microsoft's director of Windows business planning, blogged earlier this month:

"Future silicon platforms including Intel's upcoming 7th Gen Intel Core (Kaby Lake) processor family and AMD's 7th generation processors (e.g. Bristol Ridge) will only be supported on Windows 10, and all future silicon releases will require the latest release of Windows 10."

It's easy to misread that, which I suspect was intended, but Microsoft is just talking about Windows here. If you have a computer powered by a seventh-gen Intel chip or an AMD Zen CPU and you want to use Windows, Windows 10 is your only supported option.

Intel spokesman Scott Massey told us, "per Microsoft's support policy, they made the decision that Windows 10 would be the only Windows OS supported on 7th-gen Intel Core" processors. He added: "The Microsoft support change obviously doesn't impact other operating systems."

One example of Microsoft holding back support is the xHCI USB controller in sixth-generation Skylake and seventh-generation Kaby Lake; Windows 7 doesn't support that USB hardware, so

installing the operating system from a USB stick using those chips is tricky. Intel provides xHCI drivers for Windows 7 once it's up and running.

Why is Microsoft doing this?

Windows 10 must succeed at all costs. It's Windows 10 or bust. If you're buying a flash new machine, what a superb way for Microsoft to shoehorn its latest operating system onto it; that'll really help inflate its usage numbers. And it'll make life easier for its engineers: there's less hardware to test and support, less code, fewer bugs, fewer problems for everyone.

Yeah?

Yeah, I'm sure that was mentioned, oh, at least maybe once or twice internally in a meeting, that's a ballpark figure. A Microsoft spokesperson told us:

“As new silicon generations are introduced, they will require the latest Windows platform at that time for support. This enables us to focus on deep integration between Windows and the silicon, while maintaining maximum reliability and compatibility with previous generations of platform and silicon.

Windows 10 will be the only supported Windows platform on Intel’s upcoming 7th Gen Intel Core (Kaby Lake) silicon.”

Don't forget, though, Microsoft's repeatedly changed its mind on this sort of thing before.

What, when?

In January, Microsoft said it would only offer "security, reliability, and compatibility" fixes for Windows 7 and 8.1 on Intel Skylake processors until July 2017. After that cutoff point, only critical security updates would be made available – and only if they weren't a chore for Microsoft to develop and release.

After some not-so-gentle persuasion by normal folks and IT buyers, Microsoft had a rethink: it extended the release of security, reliability and compatibility updates until July 2018. Then, for its commercial customers, it changed its mind again, and pushed that date back to January 14, 2020 for Windows 7 and January 10, 2023 for Windows 8.1, when all extended support for the two operating system releases ends.

Microsoft could make similar concessions for Windows 7 and 8.1 on Kaby Lake and Zen at some point.

And Intel and AMD are cool with all of this Windows 10-only stuff?

Yeah, weirdly, seems so. You'd think they'd want their chips used on the widest possible range of operating systems to maximize sales. On the other hand, if Microsoft is pulling the plug on Windows 7 and 8.1, perhaps the chipmakers can't be bothered updating and maintaining drivers for dead-end operating systems. Perhaps they want people to stockpile today's chips.

A lot of this stuff tends to be backwards compatible, anyway. Just look at the ridiculous legacy-tangled mess that's the x86 boot process. No doubt people will find and document ways to run Windows 7 and 8.1 on the latest silicon from Intel and AMD, albeit with varying degrees of reliability. It's doable. They could port over drivers from Linux, for instance.

Microsoft loves Linux! Right?

Right.

It really will hinge on how deeply Windows 7 and 8.1 depend on particular features within Intel and AMD's CPUs. Terry Myerson, executive vice president of Microsoft's Windows and Devices Group, hinted earlier that Redmond's code in Win7, for instance, expects interrupts, power control and other hardware to be provided in ways that may differ in the latest chips:

For Windows 7 to run on any modern silicon, device drivers and firmware need to emulate Windows 7's expectations for interrupt processing, bus support, and power states – which is challenging for Wi-Fi, graphics, security, and more. As partners make customizations to legacy device drivers, services, and firmware settings, customers are likely to see regressions with Windows 7 ongoing servicing.

How convenient?

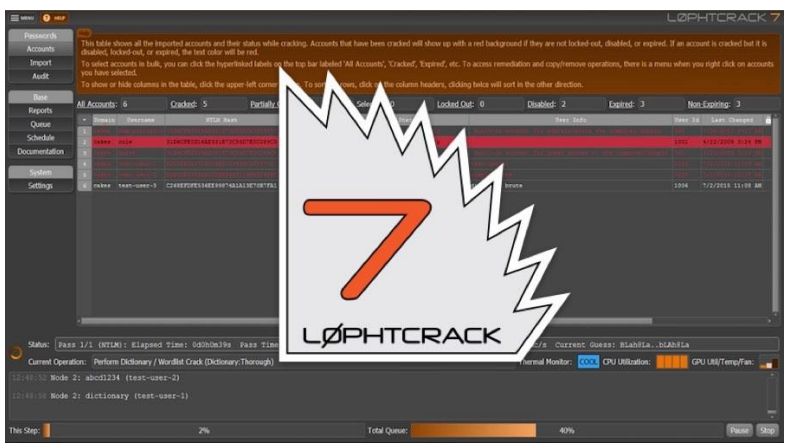
Yeah. You'd have to crack out the huge Intel and AMD reference manuals to compare the generations to see if anything really has fundamentally changed that much from, say, Skylake to Kaby Lake – and no doubt people will. I probably should.

Usually what happens is you flip a bit in a control register to enable a new mode in a feature, say, a new way of delivering an interrupt to a driver. In this example, Microsoft is trying to argue that, on the latest chips, Windows 7 can't handle this new interrupt mode, and the processor won't support the old mode, so it'll have to be emulated by the firmware or some other low-level code, which is a chore. It's a plausible excuse and also extremely convenient for Microsoft in this instance.

Basically, for now, don't buy an Intel seventh-generation Core nor an AMD Zen CPU if you want to keep running Windows 7 or 8.1. Shame. ®

http://www.theregister.co.uk/2016/09/02/windows_intel_kaby_lake_amd_zen/

L0phtCrack Updated After 7 Years, Cracks Windows And UNIX Passwords Up To 500 Times Faster



The developer of L0phtCrack, the “original” Windows password auditor, has announced the release of new and revamped L0phtCrack 7. If you use a GPU like AMD Radeon Pro Duo, compared to the previous version, L0phtCrack 7 performs up to 500 times faster. The new version also brings interface improvements and better cracking wizard.

The first version of the renowned password cracker L0phtCrack (here are some more) was released about 20 years ago. It was the first password auditing tool for Microsoft Windows operating system. It changed the way how Microsoft worked with passwords, forcing it to dump LANMAN hash algorithm and switch to NTLM.

Since then, a lot has changed in Windows and the cyber security world. Back then, L0phtCrack was able to crack the 8-character Windows password in about 24 hours. Now, L0pht Holdings, the developer of L0phtCrack, has announced the immediate availability of the revamped L0phtCrack 7. This new release comes after seven years, as no new versions have been launched since L0phtCrack 6 in March 2009.

According to L0phtCrack 7 announcement post, this password cracking tool is now 5 times faster than L0phtCrack 6. If you make use of a GPU like AMD Radeon Pro Duo, you’ll experience an amazing 500 times speedup. The new version comes with a brand new cracking engine that utilizes multi-core CPUs and multi-core GPUs.

The password cracker notes that over time, surprisingly, Windows password cracking has become easier. Wondering why? Microsoft still uses MD4, an insecure and 25 year old password hashing algorithm. On the other hand, Windows rivals Linux and OS X have offered better password hashing algorithms.

Apart from faster password cracking, L0phtCrack 7 comes with a better password auditing wizard, reporting, and scheduling. The new release works with all versions of Windows along with the support for many new UNIX password hashes.

Here is the download link for 64-bit and 32-bit versions of L0phtCrack 7.

Did you find this article helpful? Don’t forget to drop your feedback in the comments section below.

http://fossbytes.com/l0phtcrack-7-download-cracks-pc-passwords-500-times-faster/

GhostBSD 10.3 Finally Rolls Out, Offers MATE & Xfce Atop ZFS

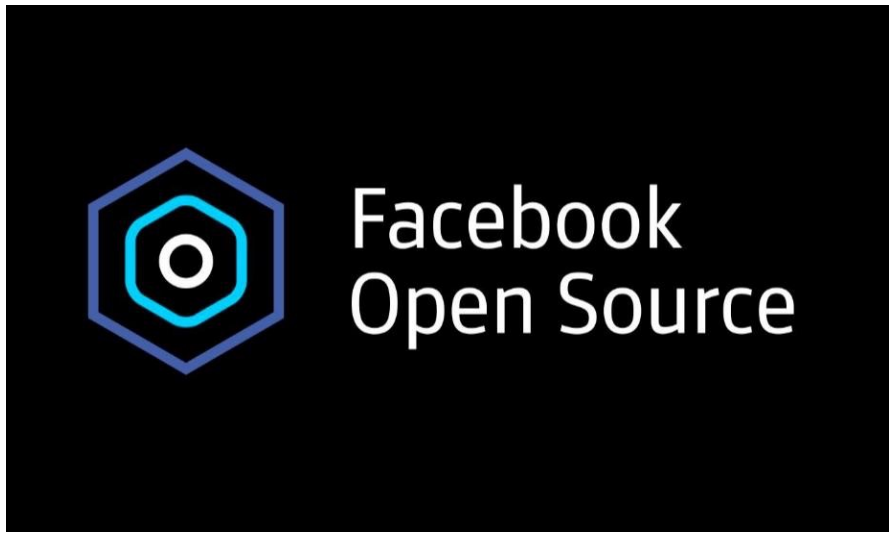
While FreeBSD 11.0 is right around the corner, today marks the official debut of GhostBSD 10.3 that in turn is based off FreeBSD 10.3.

GhostBSD 10.3 aims to be a desktop/user-friendly OS using FreeBSD. GhostBSD 10.3 adds ZFS file-system support, UEFI machine support, installer improvements, Slim as the replacement to the GDM login manager, VirtualBox support improvements, and a variety of fixes and more.

GhostBSD is planned to be updated quarterly from now on. Those interested in GhostBSD 10.3 can learn more about this operating system release at GhostBSD.org.

http://www.phoronix.com/scan.php?page=news_item&px=GhostBSD-10.3-Released

Facebook Open Sources MyRocks DB Engine And ZStandard Compression Algorithm



Facebook recently announced new open source tools named MyRocks and ZStandard at Facebook's @Scale Conference 2016. While MyRocks is a database engine that leverages all features of MySQL on the top of RocksDB database, ZStandard is a compression algorithm that can outperform current industry norms.

Many major biggies of the technology industry are slowly open sourcing their products and Facebook is no different. In May, they open-sourced CTF (Capture The Flag) hacking game platform. The Facebook CTF is a platform to host hacking challenges in Jeopardy and "King of the Hill" style Capture the Flag competitions.

Further continuing its legacy of making open source products, Facebook, at the @Scale Conference 2016, released two more open source tools for developers. Let's tell you more about them:

MyRocks

Facebook's MyRocks open source platform integrates RocksDB as a new MySQL storage engine. It lets one use RocksDB as backend storage and still leverage all the benefits of MySQL. MyRocks is brought into existence to counteract the problems of RocksDB. RocksDB doesn't support SQL layers, unlike MySQL. In MyRocks, MySQL runs on top of the RocksDB.

After deploying MyRocks into the user database (UDB) tier, Facebook would be able to save up to 50 percent storage space in comparison to InnoDB engine, which is currently used in some databases. This would also reduce the number of UDB data servers used by Facebook.

ZStandard

Another “thrilled to announce” open source project from Facebook is the data compression algorithm known as ZStandard. Facebook’s ZStandard 1.0 algorithm, available as a library and cmd tool, outperforms the compression abilities of the current industry standard algorithms by achieving higher compression ratios at twice compression speeds.

http://fossbytes.com/facebooks-new-on-the-rocks-open-source-database-engine-and-zstandard-algo/

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES
1.925.240.6652

$39.95
FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

$29.95
PC-BSD 9.1 DVD

$49.95 Save a BUNDLE!
The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

$99.95
The FreeBSD CD **or** DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

FreeBSD 9.1 Jewel Case CD/DVD $39.95

CD Set Contains:

- **Disc 1** Installation Boot LiveCD (i386)
- **Disc 2** Essential Packages Xorg (i386)
- **Disc 3** Essential Packages, GNOME2 (i386)
- **Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD $39.95
FreeBSD 9.0 DVD $39.95

FreeBSD Subscriptions

Save time and $$$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 ..... $29.95
FreeBSD Subscription, start with DVD 9.1 ..... $29.95
FreeBSD Subscription, start with CD 9.0 ..... $29.95
FreeBSD Subscription, start with DVD 9.0 ..... $29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD $29.95
PC-BSD Subscription $19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) $39.95
The FreeBSD Handbook, Volume 2 (Admin Guide) $39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) $59.95
The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 ..... $79.95

PC-BSD 9.0 Users Handbook $24.95

BSD Magazine $11.99

The FreeBSD Toolkit DVD $39.95

FreeBSD Mousepad $10.00

FreeBSD & PCBSD Caps $20.00

BSD Daemon Horns $2.00

Bundle Specials!
Save $$$

BSD Magazine
Available Monthly

FreeBSD Mall

For even **MORE** items
visit our website today!
www.FreeBSDMall.com

Lumina Desktop Customization Frameworks

by Ken Moore

The Lumina desktop environment is an open source graphical system interface for Unix-like operating systems and released under a 3-clause BSD license. Lumina is written from scratch and its design focuses on streamlining work efficiency with minimal system overhead. With the recent release of Lumina version 1.0.0, there is a lot of discussion about its flexibility and features from a user perspective. This article will instead discuss the framework within Lumina that is specifically designed for package maintainers and system administrators.

This framework enables Lumina to be easily customized for large-scale distribution, for example, within a commercial environment. In order to understand the configuration settings, we will first provide a basic overview of the various frameworks within Lumina, as it is beneficial to create a correlation between a configuration knob and “what” will actually be changed on the system.

Operating System Interface Framework

The first configuration framework within Lumina is the most foundational: the operating system interface. This does not govern which operating systems can run Lumina, but rather defines which interactions Lumina can perform with the operating system. A full chart of the current operating interfaces and their level of support is available from *https://lumina-desktop.org/get-lumina/.* This chart highlights per-operating system support for:

- System shortcuts to control panel and application store utilities
- Shutdown, reboot, and suspend support

- Battery detection and status
- Audio and screen brightness control
- CPU, memory, and disk usage statistics

TrueOS®, and by extension FreeBSD, currently have the best level of integration with Lumina, while Debian and Gentoo Linux are right behind. Any C++ developer can easily enhance and update these compatibility frameworks, as each operating system has a single file within the src-qt5/core/libLumina/ directory in the Lumina source repository. Making changes is very easy, since most options are simply comprised of a single command or library function call to run a specified operation. This design allows for the addition of new operating system interfaces as needed and provides a simple way to audit or edit operating system support as needed.

Interface Assembly Framework

The second major framework defines how the interface is assembled at runtime and many of the terms that are used during the configuration phase are defined as follows:

- Each user has their own “session” comprised of all the global options and settings that Lumina will use. Most session settings do not have a visual counterpart, but may be used to adjust multiple visual elements as necessary. For example, the locale setting will change the language used when text is displayed.
- Within a session, every physical monitor is given its own “desktop”. This desktop is always the bottom layer of the interface and is responsible for things like setting the background wallpaper or showing plugins/icons directly on the background.
- Within each desktop, there can be up to twelve “panels” attached to the desktop edges. These behave inversely to the desktop as a panel is always on the top layer of the interface, and can only be covered or hidden by a full-screen window, which is not the same as a “maximized” window. Each panel will have its own individual collection of plugins and settings.

Using these three distinctions, the system interface becomes completely customizable. The user has the capability to become as similar to other operating systems interfaces as needed, or to become something completely unique for the individual workflow.

Introduction to Configuration

Now that we have covered the important pieces of how Lumina is organized and built internally, we can start modifying those internals to suit our needs. Modification routines fall under three main categories that are directly correlated to build and distribution phases: compile, package, and post-install. Note that the earlier a modification is made, the more permanent the

setting is ingrained into Lumina. It is important to make any necessary modifications as early in the procedure as possible for your situation.

Compile-Time Modifications

The compile configuration phase is important: once Lumina is compiled, there is nothing that can be done later to affect the compiled changes, short of recompiling or reinstalling Lumina. The configuration options in this phase are quite technical and may require C++ developer experience to understand modifications to the source code, such as to enable or disable operating system compatibility options. They may also require some specialized build flag to compile-in particular default settings or files for your specific operating system. This article will not detail the entire build process for Lumina, as it is already well documented on the website, but will instead demonstrate the configuration options and what they do within the build.

The first configuration option in this phase is mostly automatic: the operating system interface selection. As soon as qmake is run on the Lumina source tree, a message displays which operating system was detected, indicating which operating system compatibility class will get used:

```
Project MESSAGE: Build OS Info: FreeBSD, amd64, FreeBSD 12.0-CURRENT
#1      ec1c56b(drm-next-4.7): Thu Aug 25 23:03:05 UTC 2016
           root@gauntlet:/usr/obj/usr/src/sys/GENERIC

Project MESSAGE: Build Settings Loaded: FreeBSD
```

If the message did not correctly detect the operating system, modify the `src-qt5/OS-detect.pri` file within the source tree and add a detection rule for that operating system. This file is a qmake file with its own special syntax rules. It is fairly easy to read and modify, but you may need to reference the qmake Manual to perform more complicated operations. This will require some C++ coding, but all changes to enable or disable operating system support options can be made in a single `src-qt5/core/libLumina/LuminaOS-<yourOS>.cpp` file. If you edit any of these files, please consider sending your changes back upstream, as the Lumina Project strives to ensure that people don't need C++ experience in order to build and use Lumina to its fullest capabilities.

The second main configuration option for this phase is the "`DEFAULT_SETTINGS`" build flag. This flag can be used to select alternate setting files or wallpapers to be used by default. These files are located within the `src-qt5/core/lumina-desktop/defaults/` directory in the source repository and represent the method by which TrueOS distributes customized system defaults. This build flag should be passed to qmake as such: `qmake DEFAULT_SETTINGS=TrueOS`.

Make sure the command is properly capitalized, otherwise it might not match the files you expect. **NOTE:** If the desired build settings do not exist within the Lumina repository, or you don't want to have your defaults publicly listed in the repository, instead modify the default setting files during the packaging phase.

NOTE 2: For a full list of instructions about how to compile Lumina from source, please view the instructions listed on the source repository: *https://github.com/trueos/lumina#how-to-build-from-source*

Packaging Modifications

The packaging phase is only required if you set one of the alternate install directory flags ("`INSTALL_ROOT`" or "`DESTDIR`") in preparation for packaging the install files to be distributed and installed on other systems. After the build or install is complete, the default setting files will be located within the staging directory configured for the build, under the share/ directory specified for that operating system($STAGEDIR/usr/local/share/lumina-desktop/) or in the etc/ directory specified for the operating system (`$STAGEDIR/usr/local/etc/`). The default wallpaper file is located in `<sharedir>/desktop-background.jpg` and may be replaced with an alternate wallpaper image as needed.

Similarly, the default settings file is located in <etcdir>/luminaDesktop.conf.dist. Modifications may be made prior to assembling the package as needed. Since it is just a text file, scripts can be used to change the file as needed.

The reason these changes are distinguished from post-install modifications is because packaging changes modify the files that the package maintains, so that when the package is installed or updated, these files will also be changed or updated automatically. However, this removes any modifications the system administrator made to them during the previous post-install phase. Also, the wallpaper file that is set here in the packaging or compile phase will be used as the default fallback wallpaper for Lumina on the system. If a new monitor is attached or enabled, this wallpaper image will be the default, making it a great way to "brand" a system with the operating system or company logo with a trivial amount of effort.

Post-Install Modifications

All post-install modifications may be performed within the `luminaDesktop.conf` file located in the operating system's etc/ directory. For example, on a FreeBSD system, this directory is `/usr/local/etc/`. Normally, only one `luminaDesktop.conf.dist` file is installed there and this file will need to be copied to luminaDesktop.conf, with all modifications made to the copied file. This is because the *.conf.dist file is managed by the package itself and any changes to it will be lost the next time Lumina is upgraded or installed.

Lumina Desktop

This file is an ASCII text file, clearly documented by comments, and uses a "`variable=value`" syntax that is easy to read and modify. Some of the available options include:

- Session settings: play login/logout chimes or enable/disable numlock on login.
- Application defaults: register particular applications as the defaults for opening files, such as the default PDF viewer, terminal, web browser, and email client. This also includes setting defaults using the mimetype registration system with a generic "`mime_default_<mimetype>`" setting.
- Theme settings: customize the Lumina theme, icon theme, default font, and fontsize.
- Desktop Interface settings: use particular wallpapers, setup a wallpaper rotation timer, auto-generate icons for items within the user's "Desktop" folder, and setup the number of panels to be used.
- Panel Interface settings: define where the panels should appear and which plugins to display.
- Favorites: setup some initial favorites for the user (applications, files, and directories). These will appear in various ways within Lumina depending on which interface plugins are used.
- QuickLaunch items: define the Start Menu using a special type of favorite specifically for the "systemstart" panel plugin.
- Generic script calling: configure an external script to run for setting up user files. This is always run after the rest of the Lumina configuration is performed, and can be used to help perform any additional user setup routines as the system requires.

This configuration file is only used when Lumina is run for the very first time by a user. It generates all of the various files and settings needed to pre-populate the user's configuration for Lumina, but will not be used again unless the user selects the option to reset their settings back to defaults from within the configuration utility. If you want to make modifications to these settings, it is crucial that you do so immediately after installing Lumina and before your users have actually run it for the first time.

Conclusions

The Lumina desktop is different from other common desktop environments in both its focus and design. By implementing a modular approach to the interface design, it becomes a trivial matter for an organization or business to implement a completely customized interface for all the systems they manage. By taking it a step further and implementing various phases of customization, Lumina makes it possible for various layers of customization to be used, ensuring that the desktop always functions properly for the given environment where it is deployed.

Lumina Desktop

More Information

Lumina desktop website: *https://lumina-desktop.org/*

Lumina Handbook*: https://lumina-desktop.org/handbook/*

qmake Manual: *http://doc.qt.io/qt-5/qmake-manual.html*

About the Author:

Ken Moore

Lumina Desktop Lead Developer

MINIX - a class-based Operating System – Part II

by Rafael Santiago de Souza Netto

This second article intends to introduce the MINIX Operating System (OS) usage in a more practical way showing aspects related with its installation, configuration and basic features, besides discussing related involved subjects.

How is MINIX distributed?

MINIX is distributed as a CD-ROM that comes together with the book when you buy it but with the Internet facilities, nowadays, we can easily and freely download it at the MINIX official website (*http://www.minix3.org*) through torrents. However, even today, I think when you buy the book you still should get media with the OS image as a bonus, probably a DVD.

In fact, MINIX's distribution image is smaller than a DVD's storage limit. For this article, I have decided to use the latest version (3.3.0). The current image bzipped has only 287MB.

After downloading its image (ISO), you need to decompress the bzipped file to finally get the real OS image (about 500MB). The nice fact about it is that it is a LIVE/CD. Then, if you burn a media with these contents, you can boot and use MINIX in a non-persistent way directly from this media.

Unfortunately, until now MINIX has no support for USB.

Installing MINIX

Today virtualization became a reality, and due to this, anyone can easily host a bunch of Virtual Machines (VMs) inside a guest environment of choice. Here, for us, the virtualization will be useful in order to make our first flight with MINIX controlled.

For this section, I will be using Virtualbox (*https://www.virtualbox.org/)* to show some details about how to boot and use the discussed OS.

You can generalize the concepts here to apply them under another Virtual Machine Manager (VMM) or even your real machine (by your own risk).*

Creating our MINIX VM

First of all, Virtualbox 3.1 is unable to run MINIX. If you are using this old version or an even older one, you should update your VMM before going ahead. For writing this part, I am using the Virtualbox version 4.1.10 and with this version I have not faced any issue running MINIX 3, in this way, no workarounds were necessary to make the virtualization possible here.

Now we are entering the part of the article where instructions based on graphical user-interfaces (GUI) are really needed. Indeed, it tends to be boring. I will try to annoy you as little as possible. So…

For creating a VM to install MINIX under Virtualbox, you need to go to the Menu's option "Machine | New". A welcome message should be shown and you should click on "Next" button. After, you should give a name for this new VM. Let's use an uncommon name here, let's be creative, let's use. "MINIX". For the "operating system type" choose "Other" and "Version" picking the combo's item "Other/Unknown".

Now you will be shown a screen asking you about base memory (RAM) size selection. I used 1GB and it worked fine for me. You should click "Next" after deciding about how much to use.

The next screen is the most important, this screen will ask you about the VM's virtual disk. Let's check the "Start-up Disk" option and the "Create new hard disk" option, too. Click on "Next". Now, for the file type, let's choose "VDI". Click on "Next" again. Hold on, almost there. Now, let's choose "Dynamically allocated" and "Next". Finally, all you should do is to define the virtual disk's size. According to MINIX's site, 1GB is sufficient, but I used 20GB. It is up to you to decide how much you need. Then click on "Next" and then "Create". Now you have a brand new and empty VM ready to install the discussed OS.

When using a VMM, you do not need to burn a DVD/CD in order to use the downloaded ISO. Usually, the VMM has the ability of "simulating" a real CD through an ISO somewhere on your well-known (even sometimes untidy) guest file-system. So, for doing it on Virtualbox, select the brand new VM in the list at the left of the VMM's GUI and click on the button "Settings". After select "Storage" and then "Storage tree", the icon related with the VM's CD-ROM/DVD. At "attributes" click on the more left "cd icon" and click on "Choose a virtual CD/DVD disk file...". It is time to indicate the decompressed MINIX image stored on your guest file-system. Also keep the "Live

*You can generalize the concepts here to apply them under another Virtual Machine Manager (VMM) or even your real machine (by your own risk).

CD/DVD” checked. Click on the “Ok” button to confirm these new settings.

Now, all you should do is to power on this VM. If everything works, you will see the boot menu depicted by Figure 1.

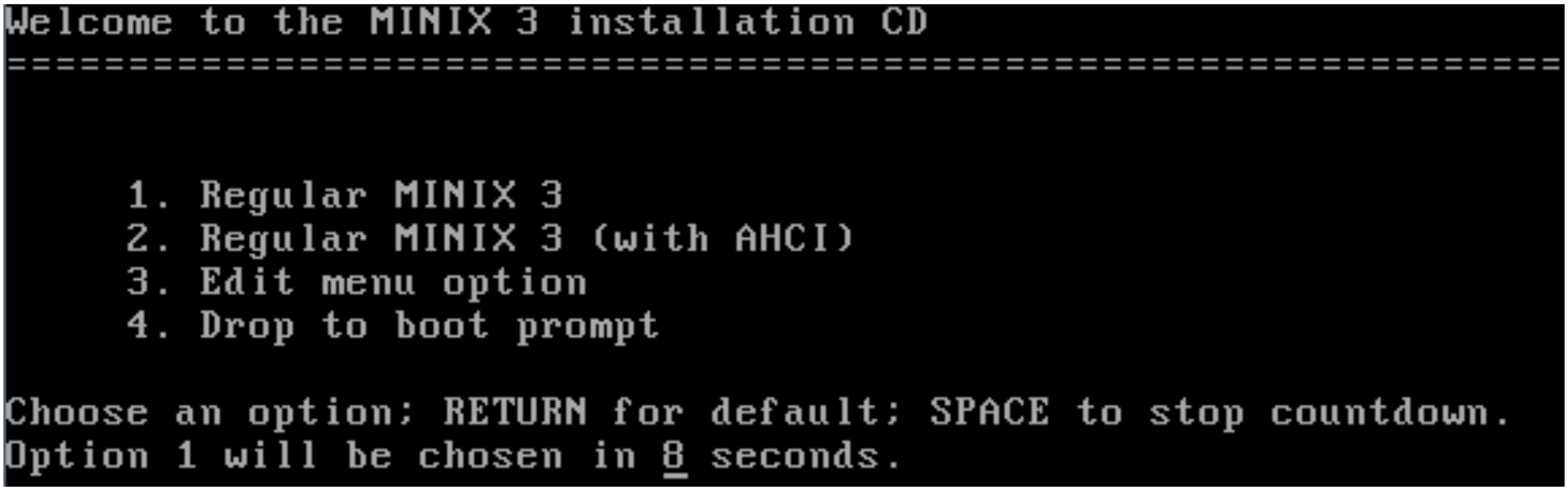


Figure 1. MINIX’s installation boot.

Some readers should recognize the NetBSD boot-loader’s “echoes” in Figure 1. You should just let the default boot happens. This will boot quickly and so you need to log in as “root” (type “root” and hit RETURN) and start the OS installing by typing “setup” and then hitting the RETURN key.

Maybe this process of logging as “root” and after calling the “setup” should be quite normal if you have already installed another non-cooked UNIX flavor. Otherwise, do not be afraid of it. Besides, remember that it is just a VM and you can easily make a new one if something went wrong. Take a look at Figure 2. In this figure, you can see the initial instructions for following up the whole OS install process.

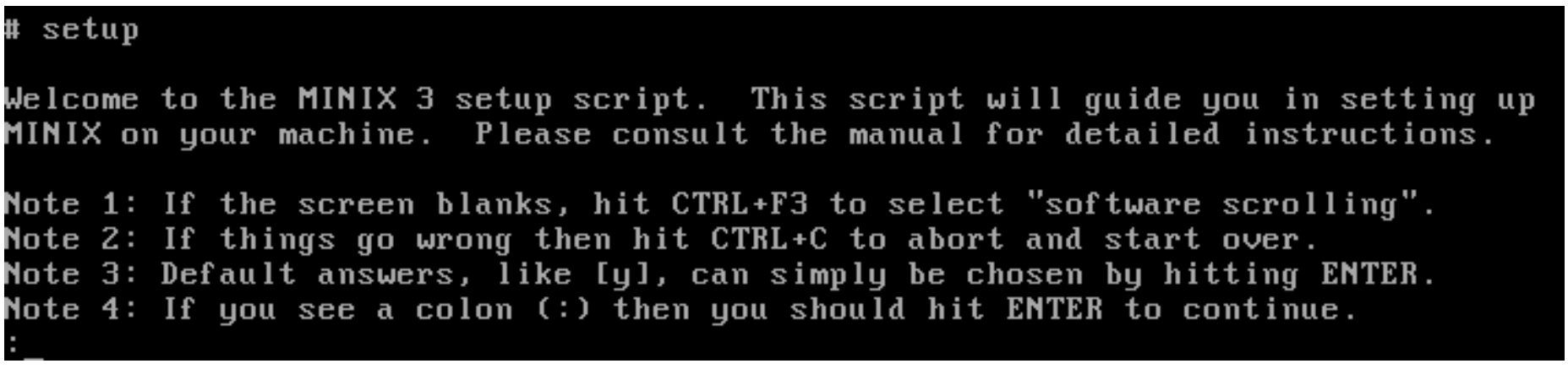


Figure 2. The initial setup instructions.

As with any install process, installing MINIX is divided into Steps. On the first Step you need to choose your keyboard’s layout. Step 2 is automatic, it just selects the full distribution installation, because we are installing it for the first time instead of updating. In Step 3, we have an important issue: disk partitions.

As we have created a VM without any previous OS installed, its disk doesn't have a valid partition, so we need to create a new one.

Fortunately, this Step is quite automated, so to begin, just hit RETURN. The setup script will seek for disks. At the end, it will list the found disks, in our case only one. The setup will automatically select the disk "0", so once again, just hit RETURN.

Then we arrive at the dangerous part (when installing it in a real machine with a previous OS). At this setup point, you need to specify the disk's region where MINIX should be installed, it can overwrite previous contents over this chosen region. Fortunately, this is not our case here because we are using a whole VM just for MINIX, so keep the "0" region selected and hit RETURN.

Hereafter, you need to explicitly confirm that you want to use the selected region for installing MINIX by typing "yes" or "no". Selecting "yes", you will be asked for the size for some directories (I use the offered defaults). Just follow the steps and the files will be copied onto your VM's disk.

The network configuration is the last Step. The installation script is smart enough to detect our Virtual Ethernet card, then, just confirm it when you are asked about it. Choose DHCP, too.

At the end of installing, all you should do is "eject" our virtual disc (Menu "Devices|CD/DVD Devices", uncheck what is checked over there) and type the command "reboot". If the VM does not reboot, you should do it manually (well, I would say virtually in this case).

Now the basic installing is done. For logging you just type "root" and confirm it hitting RETURN. Yes, it is pretty unsafe, later I will show you the way to change it.

The specific MINIX "debug" features

MINIX has specific "debugging" features. Just by executing a few commands in the terminal, it is possible to get nice reports about the Raccoon's entrails. By the way, MINIX's mascot is a Raccoon. According to the MINIX book:

> ***"The raccoon was chosen because raccoons are small, agile, intelligent, friendly – and best of all – eat bugs (at least, when there are no garbage cans available). And, being cute doesn't hurt either."***

Back to reports, these on-line reports can be useful during an Operating System class where the teacher wants to show his/her students details about a specific OS event. In Table 1, you can see a short description about these debug dumps.

An interesting aspect in the micro-kernel architecture from MINIX can be felt if you try to kill the process that has the pid "0" (init). In MINIX, when you kill the init process, the effect is merely a log-off.

Another interesting feature present in MINIX is the Reincarnation Server, a process that checks if other important processes are responding. A kind of process' watchdog. If a watched process is not responding, the Reincarnation Server (RS) creates a new process based on this previous assisted process. Then, in theory, the system as whole can be resilient against some types of failing.

The micro-kernel architecture makes it easy due to the fact of treating some important processes merely as user-space processes with few special cares. In other architectures, some processes, such as system drivers, must run in the kernel-space, and even some essential/critical processes running in the user-space have their killing disallowed or, at least, restricted. As an example, take a look at the init process on MINIX; other Unixes-likes tend to disallow or restrict this kind of operation, MINIX does not.

Key combination	Description
F1	Kernel process table
F2	System image
F4	Process privileges
F5	Boot monitor parameters
F6	IRQ hooks and policies
F7	Kernel messages
F8	VM status and process maps
F10	Kernel parameters
Shift + F1	Process manager process table
Shift + F2	Signals
Shift + F3	File system process table
Shift + F4	Device/Driver mapping
Shift + F5	Print key mappings
Shift + F6	Reincarnation server process table
Shift + F8	Data store contents
Shift + F9	Processes with stack traces

Table 1. Function key mappings for the debug dumps.

Installing new applications on our MINIX VM

Well, now we have a basic MINIX environment well-prepared. However, after executing some commands, you may start missing some applications for making your experience more pleasant. Here in this part, we will see how new software can be added to MINIX.

Before starting, a cool thing to do is to make a snapshot of the current state of the MINIX VM. Snapshots can be considered the golden rule when working with VMs; if you make a mistake, just restore the snapshot and try to find the right way once more. Never work for hours preparing a VM without taking a snapshot. Remember: this usually happens.*

The MINIX package manager

Resembling some UNIX-like flavors, MINIX also has its own package manager. This software/script tends to make the installation process of new software easier. Avoid building and installing new applications manually.

Under MINIX, this kind of useful software is known as “pkgin”.

The first time you use pkgin, it is important to update its resources. Type at the terminal the following command:

```
# pkgin update
```

If you see some pkgin’s complaint, related to Internet accessing issues, try to configure a valid DNS server. You should add to the “`/etc/resolv.conf`” a valid configuration entry. A “trick” that I like to do when I am unable to reach a valid DNS under some environment is to use Google’s DNS (8.8.8.8) or any other similar public one. You should do something like:

```
# echo “nameserver 8.8.8.8”>>/etc/resolv.conf
```

However, avoid doing it in a serious production environment, come on.

Back to the pkgin, if you use the sub-command “avail”, it will list all available packages for installing. You can get a huge list, then, it is a good choice to use pkgin with less through pipes in the following way:

```
# pkgin avail | less
```

*Sorry for the acronym ;)

For locating a specific package try the following:

```
# pkgin avail | grep "^foopkg"
```

As you can see, by the sample commands, MINIX is really UNIX-like.

pkgin's "help" sub-command can teach you much more if want to know details about other sub-tasks.

Particularly, I am not much for vi editor, I prefer the mcedit instead. This editor comes with the Midnight Commander (*https://www.midnight-commander.org/*). What about using the following command to install it?

```
# pkgin install mc
```

After running the command above, you need to do a little confirmation, then the requested package and its dependencies should be downloaded and installed. Now your MINIX VM has my favorite text editor. If you call "mcedit" at the terminal, you should get what we have depicted in Figure 3.

If you do not like mcedit and not even vi, you should try nano (pkgin install nano).

In my opinion, a useful environment should have at least basic development tools because sometimes you may need to build something on your own. I usually like to run the following pkgin install commands:

```
# pkgin install git-base-1.9.0

# pkgin install binutils

# pkgin install clang
```

Beware that depending on when you are reading this article, the package's version or even their relevance can be changed. In this case, using "pkgin avail | grep "`^pkgname`" can help or "`pkgin search <pattern>`".

*Sorry for the acronym ;)

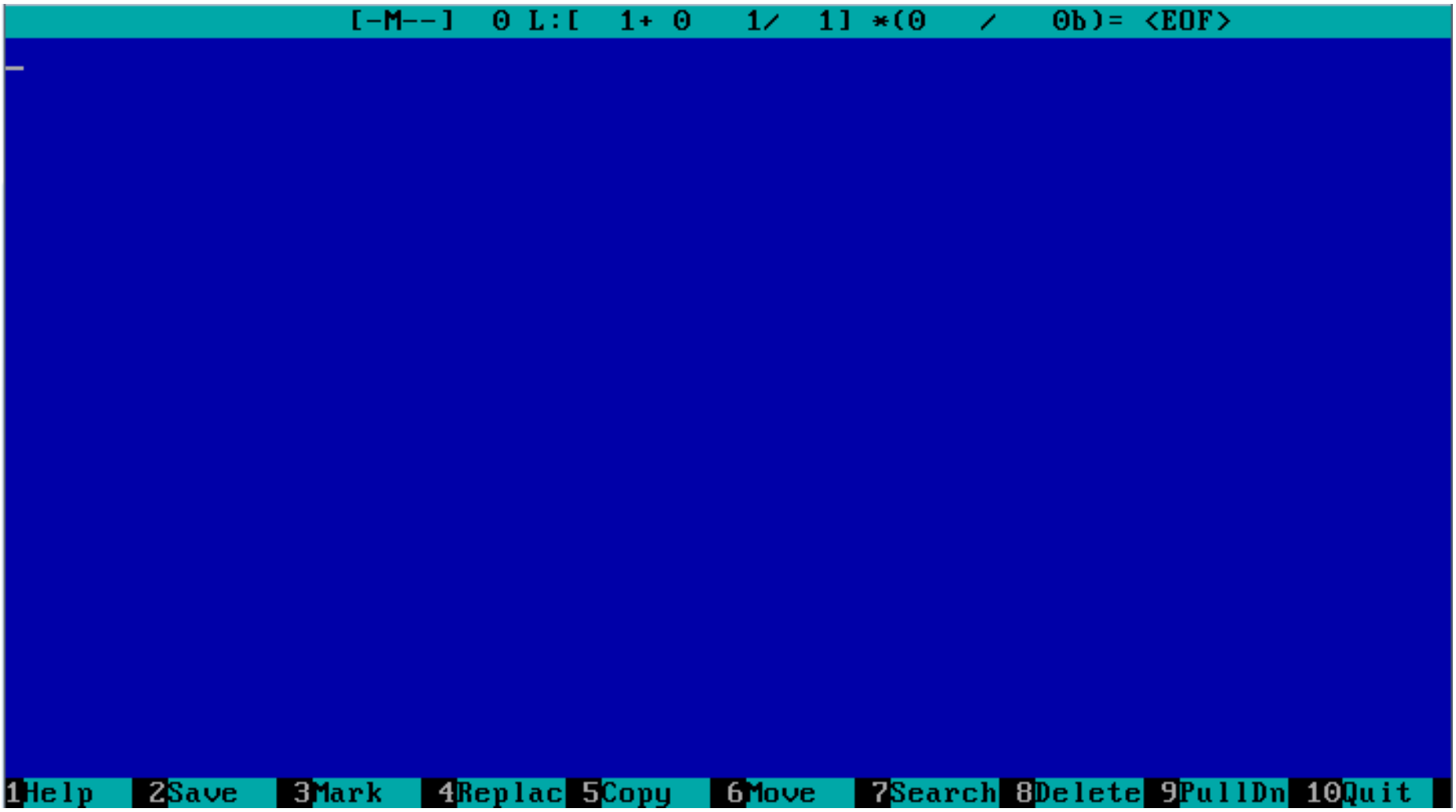


Figure 3. Some blue-screens are really nice.

I did not suggest you to install “mc” just because it brings my favorite text editor in its package but also to show you some workarounds. If you start using mc under MINIX maybe you will note some kind of annoying behavior triggered by the debug report feature present in MINIX. In mcedit, we use the Function Keys in order to access the application functions, so according to the bottom’s tip in Figure 3 you will figure out that to save data in this editor, we should use F2 key, to quit from the application F10, etc. However, you will not able to do it under MINIX unless you replace the Fn-key by ESC + number of the F-key.

The ESC + n replacement works fine at least until nine, I really do not know how to handle Function keys greater than nine. However, there is an another more elegant way to overcome this Fn-key issue. You should boot MINIX saying explicitly that the debug reports should be disabled.

To do this, you need to edit the file “boot.cfg” under the root path “/”.

By default, my "`boot.cfg`" looks like this:

```
1       clear=1

2       timeout=5

3       menu=Start MINIX 3:load_mods /boot/minix_default/
mod*;multiboot /boot/minix_default/kernel

        rootdevname=c0d0p0s0

4       menu=Start latest MINIX 3:load_mods /boot/minix_lastest/
mod*;multiboot

        /boot/minix_latest/kernel rootdevname=c0d0p0s0

5       menu=Start latest MINIX 3 in single user mode:load_mods /boot/
minix_latest/mod*;

        multiboot /boot/minix_latest/kernel rootdevname=c0d0p0s0
bootopts=-s

6       menu=Edit menu option:edit

7       menu=Drop to boot prompt:prompt

8       default=2

9       menu=Start MINIX 3 (3.3.0):load_mods
/boot/minix/3.3.0/mod*;multiboot

        /boot/minix/3.3.0/kernel rootdevname=c0d0p0s0
```

I think that yours is the same as mine. There is a nice option called "`debug_fkeys`" that you can use to inhibit the related feature. This option is boolean then "1" (the default) activates it and "0" deactivates. In this case, all you should do is replicate the content from the third line right below it adding at the end of this new line the option "`debug_fkeys=0`". It is a good idea also to use another menu's option label, let's use "Start MINIX 3 no-debug keys". The edited "boot.cfg" must be like the following code listing:

```
1       clear=1
```

```
2       timeout=5

3       menu=Start MINIX 3:load_mods /boot/minix_default/
mod*;multiboot /boot/minix_default/kernel

        rootdevname=c0d0p0s0

4       menu=Start MINIX 3 no-debug keys:load_mods /boot/
minix_default/mod*;multiboot

        /boot/minix_default/kernel rootdevname=c0d0p0s0 debug_fkeys=0

5       menu=Start latest MINIX 3:load_mods /boot/minix_lastest/
mod*;multiboot

        /boot/minix_latest/kernel rootdevname=c0d0p0s0

6       menu=Start latest MINIX 3 in single user mode:load_mods /boot/
minix_latest/mod*;

        multiboot /boot/minix_latest/kernel rootdevname=c0d0p0s0
bootopts=-s

7       menu=Edit menu option:edit

8       menu=Drop to boot prompt:prompt

9       default=2

10      menu=Start MINIX 3 (3.3.0):load_mods
/boot/minix/3.3.0/mod*;multiboot

        /boot/minix/3.3.0/kernel rootdevname=c0d0p0s0
```

After saving the “boot.cfg” execute at the terminal the reboot command. Now your boot-loader’s menu will look like Figure 4.

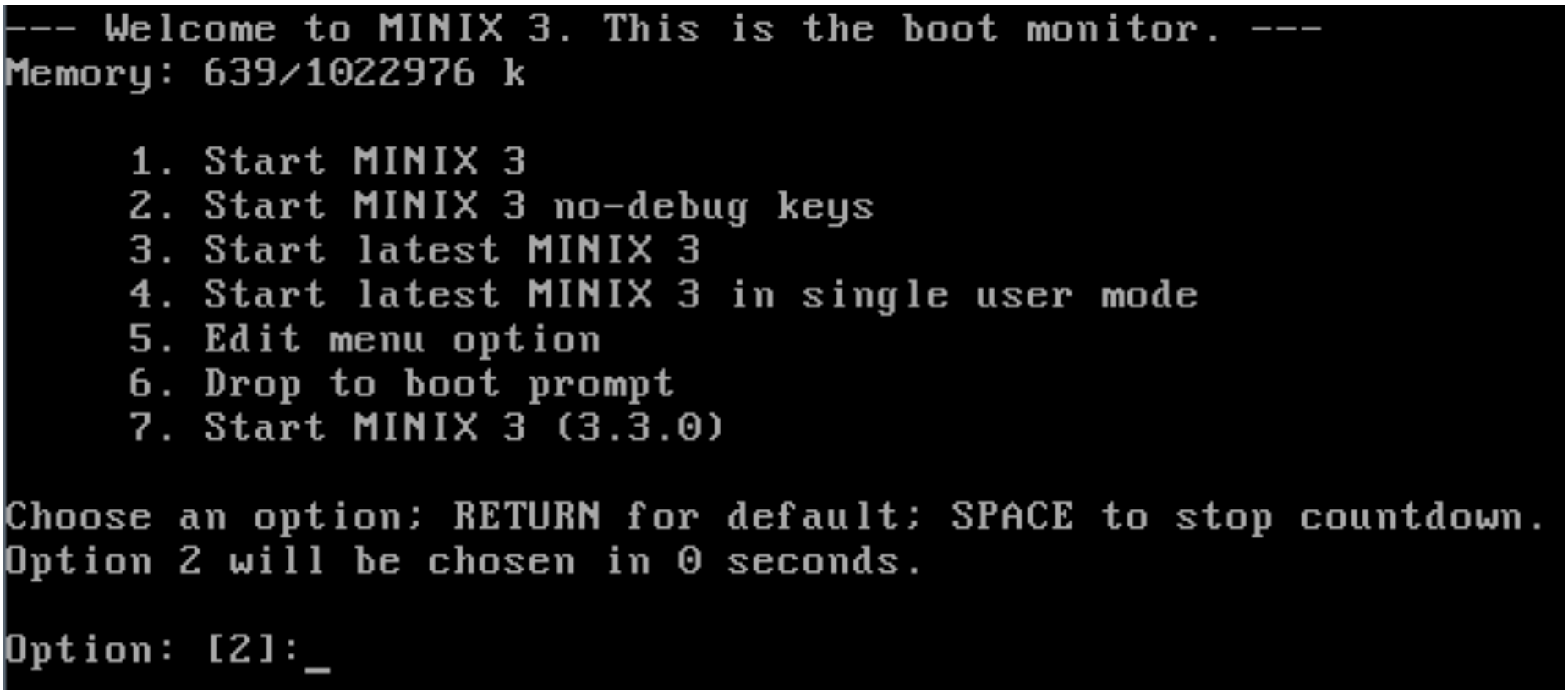


Figure 4: The boot-loader's menu showing the brand new option for booting MINIX without debugging function keys.

Now, if you choose boot option "2", the debugging function keys will be disabled and we will be able to have the Midnight Commander fully working on MINIX. You can find more boot options here: *http://wiki.minix3.org/doku.php?id=usersguide:bootmonitor*.

If you want to install the X11, you need to use an older version of MINIX, because until now X11 has not been updated for the newest OS version. Here is a good exercise for the interested reader; try to find an older MINIX version that has support for X11, download it, prepare a new VM, install MINIX and use pkgin to install X11. By the way, the command to install it is "pkgin install x11". After that, you probably can get something like Figure 5.

If you usually do not log in with run levels lower than 4 or 5, e.g.: non-graphical login, after installing X11, it is important to run the default X-Window Manager calling the script startx.

Unfortunately, until now, MINIX does not come with a true graphical web-browser, so the only choice is the browser called links, which is an improved version of the well-known text-based web-browser lynx. To install links, run the command "`pkgin install links-gui`". In order to use X11's interface instead of the terminal-based interface as depicted in the Figure 5, call the application in the xterm passing the option "-g", something like: "`links -g [<uri>]`".

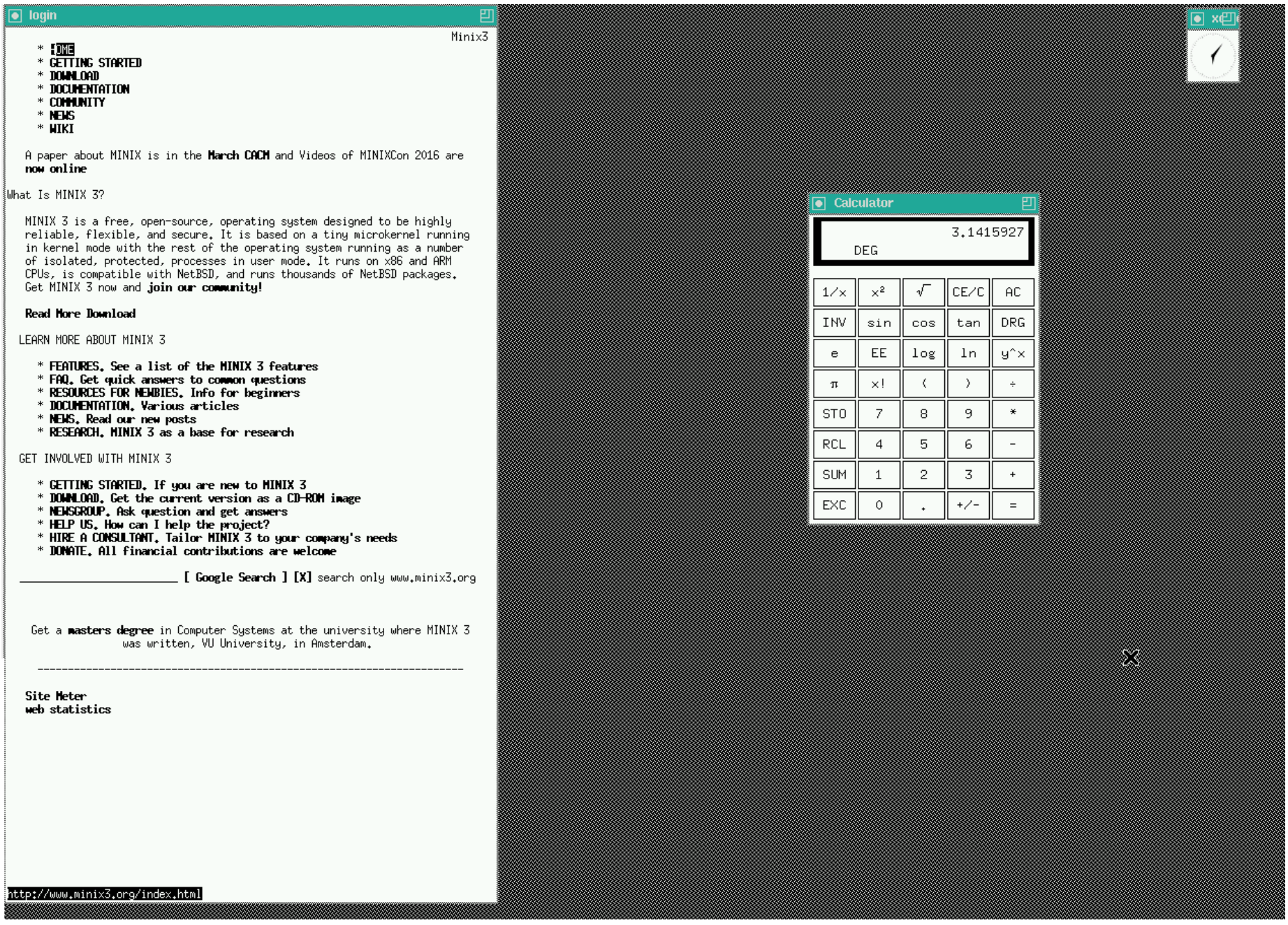


Figure 5: Here is the simple default window manager, called twm.

As I said before, by default, MINIX does not configure a password for the root user. To configuring one, it is necessary run the command “passwd” and then define a password, following the supplied instructions.

What conclusion we can get from this quick tour around the “MINIX world”?

As you have seen, MINIX is a simple, free, UNIX-like environment that started being developed in the eighties in order to be used in Operating System class’s Labs. By the way, this is the main purpose of the MINIX project even nowadays, providing to the students a simple, non-bloated Operating System to be experienced as a proof of concept. However, MINIX in some level can be used as a general purpose Operating System.

As any open-source software, the MINIX project accepts contributions. Maybe the most valuable contribution today should be related to software ports, because even being a compliant UNIX-like operating system, MINIX still suffers from the lack of some applications that could make the user experience more friendly and interesting.

Software porting can be a good software development exercise, because you are in contact with codes from other people, building issues, compatibility issues and so on. For sure, it can push you to learn new skills besides improving your previous ones.

In addition, when using some open-source system the constructive posture is a good posture to adopt. Instead of complaining about not having a version of some software of your choice under a specific platform, what about rolling up your sleeves and trying to make it possible for everybody? For sure, Open-source is endless "ant work", taking into consideration that different ants have different necessities. In this case, adding these “deltas” we can “increase” positively the final “score” ;)

Well, I also hope that this short series of articles has served to make you more interested in Operating Systems theory, MINIX and the UNIX’s philosophy as well.

You can continue diving into MINIX through its website (*http://www.minix.org*). I also really recommend the MINIX book, do not let the amount of pages intimidate you, it is quite interesting. Another interesting Distributed Operating System is the Plan 9 from Bell Labs (*http://plan9.bell-labs.com/plan9/*). If Distributed Operating Systems interests you, you can take a look at Amoeba (*http://www.cs.vu.nl/pub/amoeba/)*, this is another Operating System by Tanenbaum but in this case it is a distributed OS, and you can learn more about it by reading his book “Distributed Operating Systems”.

Of course, we also have well-known real-life Unixes-likes worthwhile code bases such as “*-BSDs”, “Linux”, etc. If you are really interested about more deep technical aspects related to Operating System Development, OSDev.org can be a good starting point (*http://wiki.osdev.org/Main_Page*). Other reading stuff that I would recommend are the “Lions’ Commentary on UNIX 6th Edition, with Source Code” and “The Design of the UNIX Operating System” books. An advanced/intermediate C Language knowledge is essential, too, if you are intending to seriously study Operating Systems. It is an essential building block.

Now if you still do not know C but you are feeling like programming in it, go ahead but please, start doing it in the right way, start learning it under a good compliant UNIX-like of your choice, creating the habit of reading the sections 2 and 3 from its man pages. In addition, do not let the “wild-pointers” make you give up, as many students nowadays tend to do or sadly are advised to do. Believe me, after really taming these pointers, you willn not be sorry, even when doing final

user-programming stuff in other cooked languages. The knowledge acquired with C Language can make you a better programmer. It probably will push you to know more about the system as whole.

Maybe the medieval prosecution of evil hidden inside of C-pointers, as well the prohibition of them, can be our “Allegory of Cave” for the Software Engineering. I find it shows us an elementary deficiency postponed from the classroom until the bookstore. Now I ask you, where does this sad recursion end? Do not let this “Segmentation Fault” occur in your programming career. Bear in mind that all types of knowledge are important for your growth, to avoid “real-mode” undefined references is the primary best practice to apply here.

About the Author:

Rafael Santiago de Souza Netto is a Computer Scientist from Brazil. His main areas of interest are Programming, Computer Networks, Operating Systems, UNIX culture, Compilers, Cryptography, Information Security, Social Coding, among others. He has been working as Software Developer since 2000. You can find him at GitHub too (as rafael-santiago) where he usually 'pushes' some of his weekend projects

Using SSD as the Level Two Cache for the FreeBSD Dual-Controller Storage Array

by Mikhail E. Zakharov

Nowadays storage systems use fast solid-state drives (SSD), not only for storing data volumes, but even as the second layer of cache. In ZFS this feature is implemented by L2ARC and it means that we can easily enable this feature on the BeaST storage system.

We do not need to describe the BeaST VirtualBox environment in detail, as it was already done many times before. Therefore we show only the configuration summary useful for the reproduction of the virtual environment:

Description	ctrl-a	ctrl-b	clnt-1
Inter-controller (private) network. Host-only adapter (vboxnet0)	IP: 192.168.56.10 Mask: 255.255.255.0	IP: 192.168.56.11 Mask: 255.255.255.0	-
Public network. Host-only adapter (vboxnet1)	IP: 192.168.55.10 Mask: 255.255.255.0	IP: 192.168.55.11 Mask: 255.255.255.0	IP: 192.168.55.20 Mask: 255.255.255.0
Base memory	2048 MB or more	2048 MB or more	Any appropriate value starting with 512 MB will do

Description	ctrl-a	ctrl-b	clnt-1
Shareable, fixed-sized virtual drives for ZFS data volumes on the SATA controller	d00, d01, d10, d11 (ada1, ada2, ada3, ada4) – each drive is 100 MB size or more	d00, d01, d10, d11 (ada1, ada2, ada3, ada4) – each drive is 100 MB or more	-
Shareable, fixed-sized virtual drives for ZFS cache on SAS controller.	f00, f10 (da0, da1) – at least 64 MB each	f00, f10 (da0, da1) – at least 64 MB each	-
System virtual drives (Dynamic-sized) on the IDE controller.	ada0 – at least 5 GB to store FreeBSD 10.3-Release default installation	ada0 – at least 5 GB to store FreeBSD 10.3-Release default installation	ada0 – at least 5 GB to store FreeBSD 10.3-Release default installation

As you can see, in addition to the previous version of the BeaST, we will only need to configure two more fixed-sized shareable virtual drives attached to the SAS controllers to emulate SSDs.

These SSDs are connected with both storage controllers, therefore in case of one controller's death the BeaST will not lose access to L2 cache through the alive controller. And as we stop L2ARC mirroring for the reasons described in the "Optimizing in-memory cache of the BeaST architecture", we also need not use ggate (GEOM Gate) interlayer for these drives. These changes in the BeaST architecture is shown in the figure below.

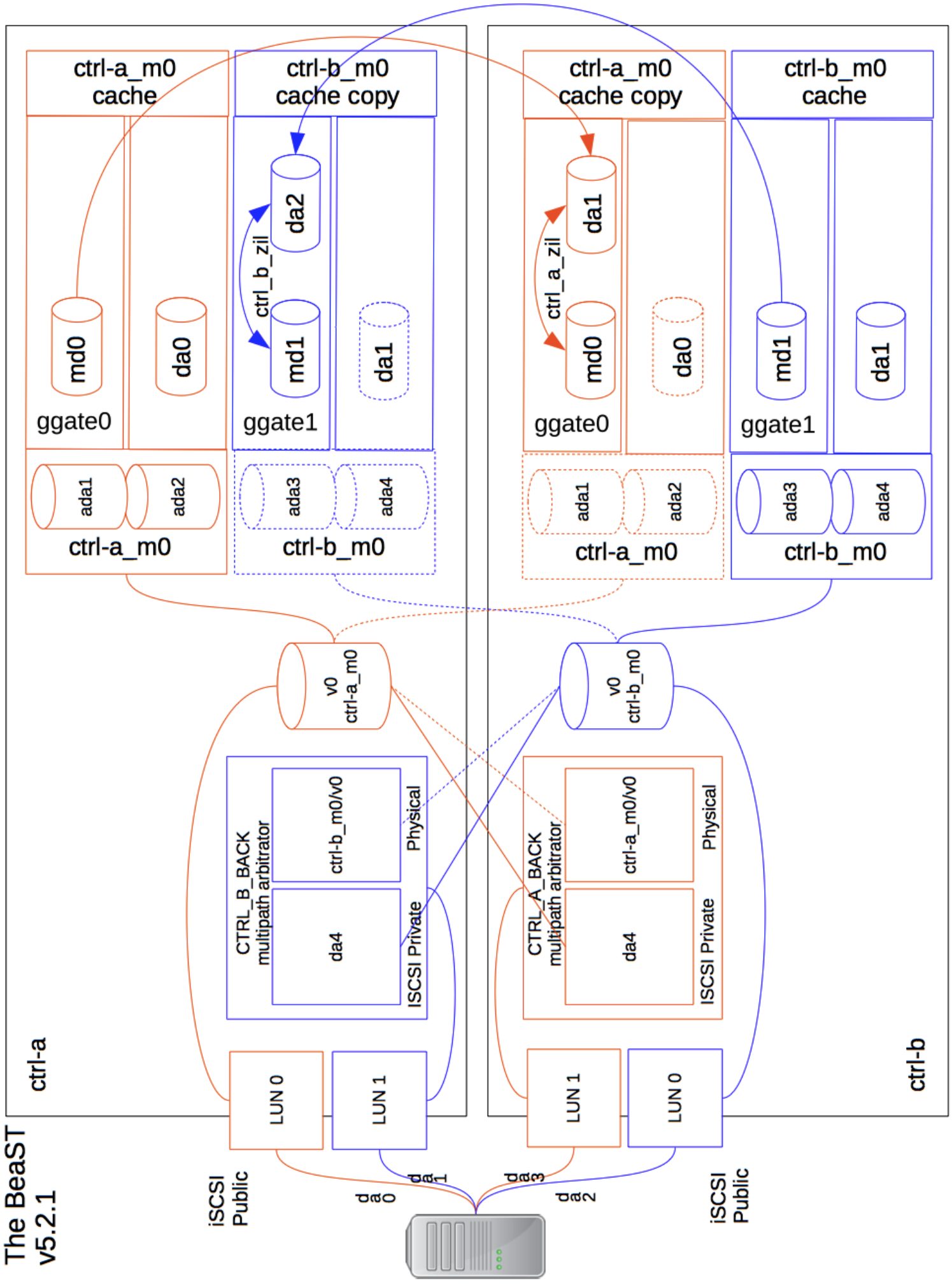


Figure 1. The BeaST architecture with cache level two cache on solid-state drives.

Now lets try to run commands to implement the fresh improvements in our architecture.

Basic preparations

As usual for our case install FreeBSD 10.3 Release on the non-shareable drives (ada0 in our case) of both virtual machines which were chosen to be the storage controllers. Typical changes in `/etc/rc.conf` for our project configuration are shown in the table below:

ctrl-a	ctrl-b

`hostname="ctrl-a"`
 	
`ifconfig_em0="inet 192.168.56.10 netmask 255.255.255.0" # Inter-controller LAN`
 	
`ifconfig_em1="inet 192.168.55.10 netmask 255.255.255.0" # Public network`
 	
`sshd_enable="YES"`
 	
`# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable`
 	
`dumpdev="AUTO"`
 	
`# VirtualBox guest additions`
 	
`vboxguest_enable="YES"`
 	
`vboxservice_enable="YES"`
 	
`# iSCSI`
 	
`ctld_enable="YES"  # Targets`
 	
`iscsid_enable="YES"  # Initiators`
	`hostname="ctrl-b"`
	
	`ifconfig_em0="inet 192.168.56.11 netmask 255.255.255.0" # Inter-controller LAN`
	
	`ifconfig_em1="inet 192.168.55.11 netmask 255.255.255.0" # Public network`
	
	`sshd_enable="YES"`
	
	`# Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable`
	
	`dumpdev="AUTO"`
	
	`# VirtualBox guest additions`
	
	`vboxguest_enable="YES`
	
	`vboxservice_enable="YES"`
	
	`# iSCSI`
	
	`ctld_enable="YES" # target`
	
	`iscsid_enable="YES"  # initiator`

Do not forget to set iSCSI “disconnection on fail” kernel variable in /etc/sysctl.conf on both systems to enable failover to the alive controller in case of disaster:

```
kern.iscsi.fail_on_disconnection=12      timeout=5
```

ZFS basic configuration

It is very well known from all our previous experiments and it is not complex at all. Therefore just create appropriate zpools and volumes:

ctrl-a	ctrl-b

`zpool create -m none ctrl-a_m0 /dev/ada1 /dev/ada2`
 	
`zfs create -V 120M ctrl-a_m0/v0`
	`zpool create -m none ctrl-b_m0 /dev/ada3 /dev/ada4`
	
	`zfs create -V 120M ctrl-b_m0/v0`

In-memory cache

Our memory drive (md0/1) to GEOM-gate (ggate0/1) map is the same as in the previous version, as we are not mirroring L2ARC drives anymore:

Memory drive	ZFS inter-layer	Controller	Description
md0	ggate0	ctrl-a	ctrl-a write-cache (ZFS ZIL) primary copy
md1	ggate1	ctrl-a	ctrl-b write-cache (ZFS ZIL) secondary copy
md0	ggate0	ctrl-b	ctrl-a write-cache (ZFS ZIL) secondary copy
md1	ggate1	ctrl-b	ctrl-b write-cache (ZFS ZIL) primary copy

And the commands to enable this structure:

ctrl-a	ctrl-b
mdconfig -a -t swap -s 128m -u 0 mdconfig -a -t swap -s 128m -u 1 ggatel create -t 1 -u 0 /dev/md0	mdconfig -a -t swap -s 128m -u 0 mdconfig -a -t swap -s 128m -u 1 ggatel create -t 1 -u 1 /dev/md1

Don’t forget to load (GEOM mirror) gmirror module as we will need it very soon:

ctrl-a	ctrl-b
gmirror load	gmirror load

Now we can prepare iSCSI targets part for the cache synchronization mechanism in the `/etc/ctl.conf` file:

ctrl-a	ctrl-b

portal-group pg0 {
discovery-auth-group no-authentication
listen 192.168.56.10
}
target iqn.2016-01.local.sss.private:target0 {
	portal-group pg0 {
	discovery-auth-group no-authentication
	listen 192.168.56.11
	}
	target iqn.2016-01.local.sss.private:target0 {

ctrl-a	ctrl-b

auth-group no-authentication
portal-group pg0
 	
# ctrl-a ZIL primary copy 	
lun 0 {
path /dev/md0
}
}
	auth-group no-authentication
	portal-group pg0
	
	# ctrl-b ZIL primary copy
	lun 1 {
	path /dev/md1
	}
	}

Then establish iSCSI connections:

ctrl-a	ctrl-b
service ctld start iscsictl -A -p 192.168.56.11 -t iqn. 2016-01.local.sss.private:target0	service ctld start iscsictl -A -p 192.168.56.10 -t iqn. 2016-01.local.sss.private:target0

And start mirroring processes:

ctrl-a	ctrl-b
gmirror label ctrl_b_zil /dev/da0 /dev/md1 ggatel create -t 1 -u 1 /dev/mirror/ctrl_b_zil	gmirror label ctrl_a_zil /dev/da0 /dev/md0 ggatel create -t 1 -u 0 /dev/mirror/ctrl_a_zil

Now we can enable caches: ZIL and both ARC/L2ARC:

ctrl-a	ctrl-b
# ZIL: zpool add -f ctrl-a_m0 log /dev/ggate0 zfs set sync=always ctrl-a_m0 # L2ARC: zpool add ctrl-a_m0 cache /dev/da0 zfs set primarycache=all ctrl-a_m0 zfs set secondarycache=all ctrl-a_m0	# ZIL: zpool add -f ctrl-b_m0 log /dev/ggate1 zfs set sync=always ctrl-b_m0 # L2ARC: zpool add ctrl-b_m0 cache /dev/da1 zfs set primarycache=all ctrl-b_m0 zfs set secondarycache=all ctrl-b_m0

Finally we must import both pools on both controllers and set zpool “failmode” variable to “continue” value in order not to stop on any failure:

ctrl-a	ctrl-b
zpool import -N ctrl-b_m0 zpool set failmode=continue ctrl-a_m0 zpool set failmode=continue ctrl-b_m0	zpool import -N ctrl-a_m0 zpool set failmode=continue ctrl-a_m0 zpool set failmode=continue ctrl-b_m0

The failover arbitrator

Lets add appropriate iSCSI target definitions to the /etc/ctl.conf:

ctrl-a

```
portal-group pg0 {
        discovery-auth-group no-authentication
        listen 192.168.56.10
}

target iqn.2016-01.local.sss.private:target0 {
        auth-group no-authentication
        portal-group pg0

        # ctrl-a ZIL primary copy
        lun 0 {
                path /dev/md0
        }

        # data volumes
        lun 10 {
                path /dev/zvol/
```

ctrl-b

```
portal-group pg0 {
        discovery-auth-group no-authentication
        listen 192.168.56.11
}

target iqn.2016-01.local.sss.private:target0 {
        auth-group no-authentication
        portal-group pg0

        # ctrl-b ZIL primary copy
        lun 1 {
                path /dev/md1
        }

        # data volumes
        lun 10 {
                path /dev/zvol/
```

ctrl-a	ctrl-b
ctrl-a_m0/v0 } }	ctrl-b_m0/v0 } }

At last assemble the arbitration construction:

ctrl-a	ctrl-b
killall -HUP ctld iscsictl -M -i 1 -p 192.168.56.11 -t iqn. 2016-01.local.sss.private:target0 gmultipath create CTRL_B_BACK / dev/da1 /dev/zvol/ctrl-b_m0/v0	killall -HUP ctld iscsictl -M -i 1 -p 192.168.56.10 -t iqn. 2016-01.local.sss.private:target0 gmultipath create CTRL_A_BACK / dev/da1 /dev/zvol/ctrl-a_m0/v0

Front-end configuration

Front-end configuration is obviously simple. Change /etc/ctl.conf to add iSCSI target information for the LUNs, accessible for client-hosts. As in all previous versions we use portal-group pg1 to enable public access:

ctrl-a	ctrl-b

portal-group pg0 {
 	
discovery-auth-group no-authentication
 	
listen 192.168.56.10
 	
}
	portal-group pg0 {
	
	discovery-auth-group no-authentication
	
	listen 192.168.56.11
	
	}

ctrl-a

```
portal-group pg1 {

        discovery-auth-group no-
authentication

        listen 192.168.55.10

}



target iqn.
2016-01.local.sss.private:target0
{

        auth-group no-
authentication

        portal-group pg0



        # ctrl-a ZIL primary copy

        lun 0 {

                path /dev/md0

        }



        # data volumes

        lun 10 {

                path /dev/zvol/
ctrl-a_m0/v0

        }

}
```

ctrl-b

```
portal-group pg1 {

        discovery-auth-group no-
authentication

        listen 192.168.55.11

}



target iqn.
2016-01.local.sss.private:target0
{

        auth-group no-
authentication

        portal-group pg0



        # ctrl-b ZIL primary copy

        lun 1 {

                path /dev/md1

        }



        # data volumes

        lun 10 {

                path /dev/zvol/
ctrl-b_m0/v0

        }

}
```

ctrl-a

```
target iqn.
2016-01.local.sss.private:target0
{

        auth-group no-
authentication

        portal-group pg0


        # ctrl-a ZIL primary copy

        lun 0 {

                path /dev/md0

        }


        # data volumes

        lun 10 {

                path /dev/zvol/
ctrl-a_m0/v0

        }

}


        }

}

target iqn.
2016-01.local.sss.public:target0 {

        auth-group no-
authentication
```

ctrl-b

```
target iqn.
2016-01.local.sss.private:target0
{

        auth-group no-
authentication

        portal-group pg0


        # ctrl-b ZIL primary copy

        lun 1 {

                path /dev/md1

        }


        # data volumes

        lun 10 {

                path /dev/zvol/
ctrl-b_m0/v0

        }

}


target iqn.
2016-01.local.sss.public:target0 {

        auth-group no-
authentication
```

ctrl-a	ctrl-b

portal-group pg1
 	
lun 0 {
path /dev/zvol/ctrl-a_m0/v0
}
 	
lun 1 {
path /dev/multipath/CTRL_B_BACK
}
}
	portal-group pg1
	
	lun 0 {
	path /dev/zvol/ctrl-b_m0/v0
	}
	
	lun 1 {
	path /dev/multipath/CTRL_A_BACK
	}
	}

At the last step is to tell ctld daemon to renew its configuration. Therefore:

ctrl-a	ctrl-b
killall -HUP ctld	killall -HUP ctld

That is all from the storage system side. We can connect now our old clnt-1 client virtual machine to test the newly created BeaST storage system. This procedure was described scrupulously in the earlier papers, therefore refer to the instructions on the BeaST project page.

Conclusions

As you can see, the difference between this and the previous version of the architecture concerns only the cache configuration. But that's one small piece of code, one giant leap for the BeaST as a whole. Putting L2 cache to SSD we can confirm that our project has the power to implement all common signs of the serious modern and reliable storage system. We will continue to develop it by adding features, software, HA, fail-back and other automation mechanisms, improving algorithms and stability. But now we are really in need of physical hardware to test our

ideas, concepts and prototypes.

And our traditional warning at the end: the BeaST is currently in the early development stage! Use it for testing purposes only! Do not implement it in production or for storing essential data, as you can lose your data!

About the Author:

My name is Mikhail E. Zakharov and I am a proud SAN/storage IBMer. 10 years of experience in large SAN and storage environments: mainly Hitachi, HP and Brocade. Empty – expect-like tool author. FreeBSD enthusiast.

FreeBSD UEFI Root on ZFS and Windows Dual Boot

by Kevin Bowling

Somehow I've managed to mostly not care about UEFI until now. On my new laptop, I decided I should give it a go. There are some small benefits, nothing life changing, but booting multiple OSes is a lot easier, especially if they are UEFI-native, and you can get a nice frame buffer the boot manager and the OS can use before starting graphically (and after, if you don't have accelerated graphics drivers).

For reference, how I run FreeBSD desktop/laptop: digital-life*

Install Windows 10 or other UEFI OS

It's easiest if you install any other co-habiting OS first. Most OS installers assume they own the entire computer, and don't let you know much about what they are really doing, especially when manipulating booting.

Windows creates a large 100MB EFI partition, plenty of room for refind and other boot loaders.

Leave free space during the installer or shrink the partition using Windows Disk Manager.

Boot into a FreeBSD 11+ live environment

We just need a live FreeBSD environment to conduct our manual install. Make sure it is 11.0 or newer for UEFI boot1 ZFS support.

The USB images with FreeBSD 11.0 and later -CURRENT snapshots have UEFI support integrated so they are directly bootable on UEFI machines. You could also use a CD/DVD or netboot.

**https://github.com/kev009/digital-life*

Enable sshd, if needed

If you want to copy/paste from this blog to the machine being installed, bring up SSH.

```
mkdir /tmp/etc /tmp/root

mount_unionfs /tmp/etc /etc

mount_unionfs /tmp/root /root

echo 'PermitRootLogin yes' >> /etc/ssh/sshd_config

passwd

service sshd onestart
```

Bring up a network interface

We'll need to grab refind during the installation.

Get a dhcp lease on your NIC or see the handbook for wireless setup*:

```
dhclient em0
```

Partition the drive

Add a couple GPT partitions. I'm doing a non-ZFS swap so I can coredump the kernel when doing FreeBSD development.

```
gpart add -a 4K -l swap0 -s 16G -t freebsd-swap nvd0

gpart add -a 4K -l zfs0 -t freebsd-zfs nvd0
```

Create a 4k aligned zpool

Standard practice these days, 4k align everything even if it's not a 4k-native disk.

Create a mountpoint and the inital zpool.

**https://www.freebsd.org/doc/handbook/network-wireless.html*

```
kldload zfs

sysctl vfs.zfs.min_auto_ashift=12

mkdir /tmp/zroot

zpool create -f -o altroot=/tmp/zroot -O compress=lz4 -O atime=off -m
none zroot /dev/gpt/zfs0

zpool export zroot
```

Boot environment compatible ZFS datasets

Nest the root dataset under ROOT so we can use boot environments in the future with beadm*:

```
zpool import -o altroot=/tmp/zroot zroot

zfs create -o mountpoint=none zroot/ROOT

zfs create -o mountpoint=/ zroot/ROOT/default

zfs create -o mountpoint=/tmp -o exec=on -o setuid=off zroot/tmp

zfs create -o mountpoint=/usr -o canmount=off zroot/usr

zfs create zroot/usr/home

zfs create -o setuid=off zroot/usr/ports

zfs create -o mountpoint=/var -o canmount=off zroot/var

zfs create -o exec=off -o setuid=off zroot/var/audit

zfs create -o exec=off -o setuid=off zroot/var/crash

zfs create -o exec=off -o setuid=off zroot/var/log

zfs create -o atime=on zroot/var/mail

zfs create -o setuid=off zroot/var/tmp

zpool set bootfs=zroot/ROOT/default zroot

chmod 1777 /tmp/zroot/tmp
```

*http://www.freshports.org/sysutils/beadm/

Perform a manual install of the distribution

This is pretty easy.

```
cd /tmp/zroot

ln -s usr/home home

tar xvJpf /usr/freebsd-dist/base.txz

tar xvJpf /usr/freebsd-dist/lib32.txz

tar xvJpf /usr/freebsd-dist/kernel.txz

chmod 1777 /tmp/zroot/var/tmp
```

Set a few things up

Set some common configurations. You may also wish to set up networking, enable SSH, etc in the `altroot rc.conf`.

```
echo 'zfs_enable="YES"' >> /tmp/zroot/etc/rc.conf

echo 'dumpdev="AUTO"' >> /tmp/zroot/etc/rc.conf

echo 'powerd_enable="YES"' >> /tmp/zroot/etc/rc.conf

echo 'sendmail_enable="NONE"' >> /tmp/zroot/etc/rc.conf

echo 'zfs_load="YES"' >> /tmp/zroot/boot/loader.conf

echo 'kern.geom.label.disk_ident.enable="0"' >>
/tmp/zroot/boot/loader.conf

echo 'kern.geom.label.gptid.enable="0"'  >>
/tmp/zroot/boot/loader.conf

printf "/dev/gpt/swap0\tnone\tswap\tsw\t0\t0\n" >> /tmp/zroot/fstab

tzsetup -C /tmp/zroot

chroot /tmp/zroot/ passwd
```

Install refind

UEFI has lots of bells and whistles. We're going to use the refind boot manager. I'm relying on the "fallback" efi loader, `bootx64.efi`. You may need to toggle things around in your system's firmware for that to work, or teach the EFI NVRAM about refind*. See the refind site for more details.

```
cd /tmp

fetch http://downloads.sourceforge.net/project/refind/0.10.3/refind-bin-0.10.3.zip

unzip refind-bin-0.10.3.zip

rm refind-bin-0.10.3.zip

mkdir /tmp/efi

mount_msdosfs /dev/gpt/EFI%20system%20partition /tmp/efi/

cd /tmp/efi/EFI/Boot

mv bootx64.efi bootx64-windows-10.efi

cp /boot/boot1.efi bootx64-freebsd.efi

cp -a /tmp/refind-bin-0.10.3/refind/icons .

cp -a /tmp/refind-bin-0.10.3/refind/refind_x64.efi bootx64.efi

cp /tmp/refind-bin-0.10.3/refind/refind.conf-sample refind.conf
```

As good hygiene, you might consider updating `bootx64-freebsd.efi` whenever a point release is done. You could also keep an eye out for refind updates. This is easily done since it's just a DOS filesystem.

Configure refind and add menu entries

Set the values of timeout, and scanfor to manual to speed things up a bit in `refind.conf`.

Then add a couple entries:

```
cat << EOF >> refind.conf
```

*http://www.rodsbooks.com/refind/

```
menuentry "FreeBSD/amd64 -CURRENT" {
    loader \EFI\Boot\bootx64-freebsd.efi
    icon \EFI\Boot\icons\os_freebsd.png
}

menuentry "Windows 10 Professional x64" {
    loader \EFI\Boot\bootx64-windows-10.efi
    icon \EFI\Boot\icons\os_win.png
}
EOF
```

Finish, reboot and enjoy!

That's it. Unmount the efi partition and reboot.

```
cd
umount /tmp/efi
reboot
```

You should be greeted by refind, otherwise take a look through your firmware boot order and make sure the firmware nvram for Windows Bootmanager isn't first.

GELI

Keep an eye out for GELI full disk encryption* on top of ZFS on root.

Don't forget to do the swap partition as well.

http://kev009.com/wp/2016/07/freebsd-uefi-root-on-zfs-and-windows-dual-boot/

**https://ericmccorkleblog.wordpress.com/2016/05/28/freebsd-geli-support/*

Thanks to

Most of this was cribbed from the following sources:

- Eric McCorkle, Steve Hart and others for adding ZFS boot and a ton of other improvements (GELI) to the FreeBSD UFI loader.
- Trond Endrestøl's blog, for mentioning refind and the overall UEFI landscape on FreeBSD.
- Calomel, for a decent overview of manual ZFS on root installation.
- /usr/src/usr.sbin/bsdinstall/scripts/zfsboot for some ZFS specifics.
- An imaging script my colleague Jason Wolfe did for ZFS and boot envs at work.

About the Author:

Kevin has been using FreeBSD professionally for a few years at Limelight Networks and has over a decade of both software development and high scale systems engineering experience on other UNIX variants.

OpenBSD in 2016: The Year of the Secure Desktop

by David Rodriguez

One corporation has controlled the desktop for quite a while; perhaps it's that friendly start menu that so many are accustomed to? Trends over the past decade show a great shift in computing. The iPhone changed the way most people access the internet. With the massive adoption of Android, Linux has become the most widely used user-agent on many websites.

With the change in direction over the past few years, and Windows 8's tablet/touch oriented interface, it seems that M$ has shifted focus toward mobile computing. With smart phone sales soaring and PC sales dwindling, the largest desktop OS manufacturer is leaving a massive gap ripe for a secure, modern OS to fill. A gigantic puffer fish waddles in from stage left dropping off a fresh copy of 5.9. And here is where our story begins…

The door is wide open for non-MS options on the desktop, and FOSS developers have been working to fill that gap for quite sometime. FOSS has come a very long way in its general usability, the feature set has easily reached parity with (and surpassed in certain situations) closed source software. OpenBSD in many ways takes Free and Open Source to the fullest extent. One of the OpenBSD community's strongest beliefs is that all software should be Open Source. Even drivers are no exception, binary blobs* simply are not accepted. This is a strong foundation on which to build a secure, stable and simple to use desktop OS.

OS Security is an increasing trend

The OpenBSD developers, like many operating system developers these days, take security into consideration during design and implementation. Nearly all of the commonly used operating systems take steps to actively improve security. Some of those steps include: having large security

**https://en.wikipedia.org/wiki/Binary_blob*

teams, performing code audits, responsible disclosure programs with bug bounties, using the most current ciphers and secure pipelines for code check-in, to name a few. Although many of those items can help to improve security, OpenBSD provides something that few other operating systems can provide, the source code in its entirety. The source code is arguably the most important feature of a secure operating system as it allows for the most complete auditing of the software. Having access to the source code certainly does not guarantee it to be free of bugs or vulnerabilities but, in a practical sense, it means you have everything you need to find backdoors in the software regardless of whether they were placed maliciously or accidentally.

The OpenBSD developers have a track record of making security oriented decisions regardless of the initial fallout to users. Big changes to security can break certain applications; it can be easily said that OpenBSD has helped (or forced at times) to move software in a more secure and positive direction on many occasions. In some ways this could be viewed as a negative, in my opinion it's a tradeoff that needs to be taken into account when selecting an OS. OpenBSD developers have taken aggressive steps in the past to improve security and will continue to do that in the future, in my opinion that is only a positive. The very nature of software (operating systems, applications, libraries, etc.) is to change over time as new challenges and needs will continue to arise.

Modern Desktop in 2016

In order to be considered a great desktop OS, we have to meet a few basic criteria:

1. Packages and repositories that are well maintained.
2. Modern web browsers.
3. Capable of streaming media.
4. Applications to perform basic office tasks (written documents, spreadsheets with formulas, presentation software, etc.).
5. Software to connect to the Internet.
6. Print documents, images, spreadsheets, etc.

OpenBSD definitely meets the criteria of a modern desktop OS, but the path to get there can be a bit rocky. As an OpenBSD user, you must be ready to read and understand extensive documentation, have a willingness to overcome challenges and a desire to say no to the default OS. The OpenBSD installation poses a few challenges and the order of operation is especially important. If you’re security oriented, you likely use full disk encryption on laptops and desktops, so don't forget to use 'bioctl' to encrypt your disks before starting the install.

Choosing a secure Desktop OS

If you are the type of person that is very security oriented and completely paranoid, you have a few options in choosing your OS: write your own OS, audit the source of a FOSS distro and/or find a group that is trustworthy. I think the ideal scenario is one when the individual does all three, contributing to an OS, while auditing the code of others in a trustworthy group. Many OS's do not provide source at all, and others frequently include device drivers, referred to as 'binary blobs' (closed source software).

A proven track record leaning towards security

OpenBSD is freely available for download with no requirement to purchase a license, allowing OpenBSD developers to make incredibly tough decisions that could send many paying customers running. OpenBSD developers have a track record of enabling security by default. Most notably, Theo de Raadt has said that security controls that can be disabled are meaningless, because they will be disabled. The decision to turn on ProPolice (stack protection) caused issues with many applications, but it also forced the hand of many software developers to move forward.

One of OpenBSD's newest security features, “pledge”, helps to mitigate the effects of bugs (potential vulnerabilities). If you've never heard of pledge() take a minute to read about it or watch a few videos by Theo de Raadt (see below). An application can pledge that it will use a specific set of system calls. If it were exploited, it would be terminated with a SIGABRT as soon as it attempted to make other system calls. This is not a cure all, but can be a vast improvement over other exploitation mitigation strategies that are frequently disabled or are too complicated to use (SELinux, Capsicum or systrace). Analysis performed by the OpenBSD group showed that the greater majority of programs used a rich set of system calls during initialization(), then uses a much smaller set once processing happens in the main() loop. If the application's use of pledge() occurs after initialization, and before the main loop, we have an easy to use system to mitigate the effects of the majority of application attacks.

Get ready for pleasant surprises

Everything just seemed to work after the install was finished as I was lucky enough to have a Lenovo T430 for the install. I didn't have to download drivers, or perform any of the ndiswrapper magic of years past, it just worked. Starting with the latest version of OpenBSD 5.9 on a flash drive, I went to work. After a few minutes, I had a working desktop. Screen resolution looked good, wired and wireless network devices were working, and most of the laptop’s features worked. I was able to setup multiple monitors easily and the card reader worked without issue. Having done a bit more research, it seems this is because the OpenBSD developers tend to eat their own dog food, so to speak. Most of them run OpenBSD on their systems, both desktops and laptops, so it does get quite a bit of use outside of the server racks in the data center. Having per-

formed several hundred OS installs throughout my life on many different types of hardware, I can say it's very refreshing to have so many things work right out of the box.

When you're ready to get started on work, an OpenBSD desktop can support you. OpenVPN installed as a package 'pkg_add openvpn' is the simplest way to get remotely connected to your office. OpenBSD serves many organizations very well as a firewall or VPN appliance, so it's right at home connecting to VPNs (SSLVPN, IPSEC) from the client perspective. LibreOffice takes care of all your document needs, plus you have numerous web browsers and development IDEs to choose from. Having both Ruby and Python kept up-to-date, as packages, in OpenBSD is icing on the cake.

I used XFCE as the desktop environment, it's light weight and simple (an important consideration when using a 4+ year old laptop). It may not have all the features of KDE or gnome, but I enjoy that it's so light weight. OpenBSD has thousands of packages available for easy install. You have several quality options available in the repo; KDE, Enlightenment and gnome, to name a few. It's pretty simple to install the desktop environment of your choice. After the OpenBSD install completes, you'll be prompted to reboot. If you selected the option to run 'X' on startup you'll land on a fluxbox desktop, otherwise you'll be staring at a terminal.

The OpenBSD install is unlike most OS installers today, in many ways. First off, it's very fast, usually done in less than five minutes. Second, the result of the initial install will not have you land on a nice looking GUI desktop. Third, it's small in size, at only 223M for the ISO, the download is pretty fast even on slower DSL connections and public wifi. Fourth, you have many options to verify the validity of the download. Although the install process is not as simple as PC-BSD's or uBuntu's, you have everything you need to get a modern desktop in a few steps with the cli.

Steps to success

Before starting the install, there's a quick decision to make. Do you want to encrypt your disk? If so, use the 'Shell' to get started. You'll want to setup partitions (fdisk and disklabel), encrypt a disklabel partition and setup swap space outside of the encrypted disklabel partition, if needed. OpenBSD swap is encrypted by default, many choose not to place a swap partition in the encrypted disklabel partition, but I'll leave that decision up to you.

If you want to encrypt the disk, hit “s” to drop to a 'Shell':

```
======Begin Disk Encryption steps======

Welcome to OpenBSD/adm64 5.9 installation program.

(I)Install, (U)Upgrade, (A)Autoinstall or (S)hell. S
```

Use 'fdisk' to initialize the disk:

```
# fdisk iy sd0

Writing MBR offset 0.
```

Use 'disklabel' to setup partitions:

```
# disklabel E sd0
```

Hit 'p' to check the current status of the partitions:

```
Label editor (enter '?' for help at any prompt)

> p

OpenBSD area: 6441929650; size: 41929650 free: 41929650

# size          offset     fstype [fsize bsize cpg]

c: 41929650     0          unused

>
```

Setup a swap partition. (Optional):

```
> a b

offset: [64]

size: [41929650] 2g

FS type: [swap]

>
```

Create a 'RAID' Partition. (This is the partition to be encrypted):

```
> a a

offset: [4209030]

size: [37720620]

FS type: [4.2BSD] RAID
```

Write and Quit disklabel:

```
    •   w

    •   q

No label changes.

#
```

Encrypt the disklabel partition:

```
bioctl -c C -l /dev/sd0a softraid0

sd1 at scsibus1 targ 1 lun 0: <OPENBSD, SR CRYPTO, 005> SCSI2 0/
direct fixed sd1: 19445MB, 512 bytes/sector, 39824607 sectors

softraid0: CRYPTO volume attached as sd1

# exit

======End Disk Encryption steps======
```

Make sure to take note of the location of the 'Crypto volume' output highlighted in Blue. Now that we setup our encrypted disklabel partition, we can continue with the process of a normal install. After exiting the 'Shell', we are brought back to a familiar prompt. Hit 'I' to start the install.

```
Welcome to OpenBSD/adm64 5.9 installation program.

    (I)Install, (U)Upgrade, (A)Autoinstall or (S)hell. I
```

From this point on, the OpenBSD install is very simple and I'll skip many prompts as I'm sure you can figure out your host name, user name and network config. The defaults work very well for most situations. Make sure to select the correct drive for the install, the default will likely be incorrect in this situation (if you chose to complete the disk encryption steps above).

```
Available disks are: sd0 sd1.

Which disk is the root disk? ('?' for details) [sd0] sd1

No valid MBR or GPT.

Use (W)hole disk MBR, whole disk (G)PT or (E)dit? [whole]
```

It's beyond the scope of this article to discuss the various options for disk partitioning in OpenBSD as the subject is very well documented. I suggest you head directly to the source and read the OpenBSD documentation on Disks and Partitioning*.

You'll be presented with the list of default sets.

```
Select sets by entering a set name, a file name pattern or 'all'. Deselect sets by pre-
pending a '' to the set name, file name pattern or 'all'.

Selected sets are labeled '[X]'.

[X] bsd         [X] base59.tgz    [X] game59.tgz      [X] xfont59.tgz

[X] bsd.rd      [X] comp59.tgz    [X] xbase59.tgz     [X] xserv59.tgz

[X] bsd.mp      [X] man59.tgz     [X] xshare59.tgz    [X]

Set name(s)? (or 'abort'       or 'done') [done]
```

After the sets are installed, add one minor tweak before rebooting:

```
CONGRATULATIONS! Your OpenBSD install has been successfully completed! To boot
the new system, enter 'reboot' at the command prompt.

#

# sed 's/rw/rw,noatime,softdep/g' /mnt/etc/fstab > /mnt/new_fstab
# mv /mnt/new_fstab /mnt/etc/fstab
```

https://www.openbsd.org/faq/faq14.html

That's it for the initial install.

```
# reboot
```

There's so many options for desktops environments, I'm sure you'll be able to find a set of packages that meet your needs. Before you start installing packages, make sure to setup your “$PKG_PATH”, and also add it to your '.profile'. Have a look at this OpenBSD documentation to identify your closest mirror.

```
#   export

#   PKG_PATH=ftp://ftp.openbsd.org/pub/OpenBSD/5.9/packages/amd64
```

In OpenBSD, you have two general options to acquire and install software, Packages and Ports. Packages are pre-compiled binaries ready to run. Ports offer you the option to compile the binaries yourself and modify any configuration options you see fit. Depending on the type of user you are, you may be able to stick with packages only. When you stick to the packages, updates are as simple as running “pkg_add -u”. The most common reason I find myself using a port is because of a licensing issue. Most software doesn't meet the strict requirements of the BSD license and requires user interaction to be installed. as simple as running “pkg_add -u”. The most common reason I find myself using a port is because of a licensing issue. Most software doesn't meet the strict requirements of the BSD license and requires user interaction to be installed.

The process to install a port is rather simple, it typically involves running no more than 'make && make install' to compile and install the port. Depending on why you chose to use a port will determine how much configuration is necessary. Let's say you need to install a version of Apache with a standard feature removed, or a version of Nginx with a special feature included, the port will allow you to customize, compile and install the software to your exact specs. It can be a tough decision to use a package or a port, my general rule of thumb is that packages are the better way to go because the total install process is faster and it's very simple to upgrade all packages at the same time, whereas ports offer more flexibility, if required.

Challenges you'll face down the road

The one device that didn't work correctly was the fingerprint reader, everything else worked well on my Lenovo T430. Working with OpenBSD hardware can become an issue. If you have older systems laying around, pretty much any common desktop/laptop made in the last 15 years, you're in luck! Most things will just work. The newest hardware may present problems, if a driver is not readily available, the challenge of writing one could be daunting, if not impossible all together.

**https://www.openbsd.org/ftp.html*

When it comes to choosing your hardware, you'll want to be very selective before starting the install. OpenBSD provides excellent documentation on this topic . If you've got an amd64, i386, you're good to go. If you've got an arm processor, you may have issues; PandaBoard, BeagleBone, BeagleBoard are supported, but Raspberry Pi is not.

OpenBSD is ideal in so many ways, but certain applications only run on Windows. Virtualization provides an answer to the challenges that OpenBSD will not solve. Using software like Wine may be familiar to Linux users. Having been a user of Wine for over a decade now, I can say that it has failed to run applications correctly in more than 60% of the instances I've attempted to use it. Rather than head the direction of application virtualization (e.g. Wine), I prefer to virtualize the OS to really keep the application isolated. If you've worked with applications developed in-house or hardware that is reliant on a specific version of out dated Java/Flash you understand why it's important to keep certain types of applications as isolated as possible. Qemu allows for a full installation of Windows should you have a requirement to run an application written specifically for Windows only.

Virtualization solves many challenges, but does not solve them all. A second hard drive is an excellent choice if you are a gamer and need the performance of running on bare-metal. Much of the gaming found on OpenBSD is not up to spec when compared with modern PC and console games. Until Blizzard writes an OpenBSD version of Overwatch, you are somewhat limited in your gaming options.

Connecting to multiple wireless networks - depending on the desktop environment you select, you may or may not have access to a simple UI for easily connecting to wifi hotspots. Since I only connect to a few wifi networks, I found scripting my way out of this challenge to be a fun process. If you are looking for assistance with connecting to multiple wireless, this guide will help. The cli can be a fun option, but kde or gnome desktops can greatly help the experience if a wifi GUI is a requirement.

Extensive printing needs may be a large hurdle to jump, and could result in a trip to Amazon for a more compatible model. Printing can be accomplished with CUPS, so most Linux users will be familiar with the process. If you're printing numerous documents on a daily basis, I'd suggest checking compatibility of your printer before making the switch to OpenBSD. Many printers these days can easily help to solve the problem using network connectivity or a slot for a memory card. I successfully printed using an off the shelf printer acquired at a local retail store with just a few minutes of setup time.

Conclusion:

With all things considered, I really enjoy my experience with OpenBSD as a desktop, it's reliable and functional, but it's not for everyone. OpenBSD provides the type of customizable experience I look for in a secure desktop.

OpenBSD

Many Linux distros I've installed in the past five years make assumptions that sacrifice security for the wrong reasons. There are numerous operating systems available today, and the choices can vary widely. I won't try to convince anyone that OpenBSD is correct for them, but I find that it fits my needs perfectly. With the release of OpenBSD 6.0 right around the corner, we've got lots to look forward to. Performance improvements to the iwn driver and 802.11 wireless, security improvements, like W^X is now strictly enforced by default and numerous patches to LibreSSL and OpenSSH. There are numerous reasons to look forward to the release of OpenBSD 6.0, what are you looking forward to?

One Final Thought… I'd like to give an enormous Thank You to Theo de Raadt and the rest of the OpenBSD team, you've produced a wonderful operating system.

OpenBSD sites and support:

- *http://www.openbsd.org/faq/faq1.html*
- *http://www.openbsd.org/faq/faq4.html*
- *http://www.openbsdsupport.org*
- *http://www.openbsd.org/60.html*
- *https://www.openbsd.org/papers/hackfest2015-pledge/mgp00001.html*
- *https://www.openbsd.org/papers/hackfest2015-pledge/mgp00001.html*

OpenBSD Desktop:

- *http://www.bsdnow.tv/tutorials/the-desktop-obsd*

OpenBSD Full disk encryption:

- *https://www.bsdnow.tv/tutorials/fde*

About the Author:

David Rodriguez - Sr. Systems Engineer

https://www.linkedin.com/in/david-rodriguez-5a90768b

FREENAS MINI
STORAGE APPLIANCE

IT *SAVES* YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

Example of one-bit corruption

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time*.**

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured

iXsystems™

http://www.iXsystems.com/mini

intel inside™ ATOM™

FREENAS CERTIFIED STORAGE

With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- Custom built and optimized for your use case
- Installed, configured, tested, and guaranteed to work out of the box
- Supported by the Silicon Valley team that designed and built it
- Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**

FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

http://www.iXsystems.com/storage/freenas-certified-storage/

FreeNAS Getting Started Guide: Part 4, Plugins

by Mark VonFange

This article series is intended to serve as an introductory guide to assist FreeNAS users in planning, installation, configuration and administration for their FreeNAS storage systems. This month's article will cover installing and updating plugins in FreeNAS 9.x.

The FreeNAS Plugin System

The FreeNAS web interface comes with an intuitive, self-contained menu for installing plugins for many popular and useful third party applications such as Plex, OwnCloud, BTSync, Crashplan and many more. To get to the plugins menu, simply click on the "Plugins" Icon on the top bar. This will bring up a list of available plugins for installation (Figure 1).

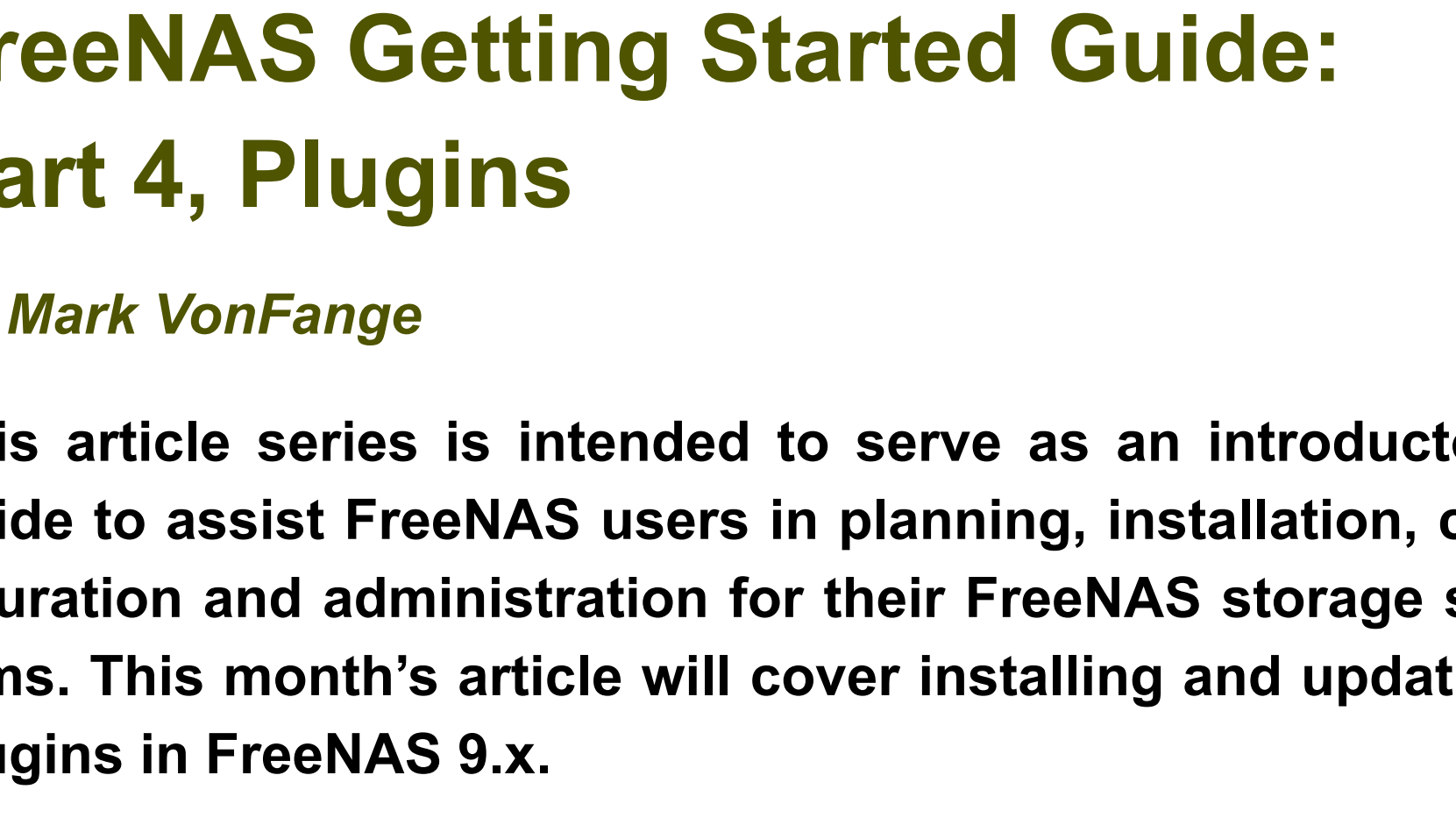


Figure 1 - FreeNAS Plugins Menu.

To install a plugin, simply click on the row of the desired application and then click on the “Install” button that appears at the bottom of the menu screen (Figure 2). This will bring up a pop-up menu with “OK” and “Cancel” options. Clicking the OK button will initiate the download and installation process. Do not navigate away until the progress bars show the process to be complete and the pop-up menu disappears, otherwise the installation will be unsuccessful and you will have to start the process over.

You can also install .pbi applications that aren’t in the populated repository list by clicking on the Upload Icon, which brings up a pop up menu. Simply click on the Browse button, select your download file, then click on the “OK” button to initiate the installation process. Your plugin will appear in the Installed tab once the process is complete.

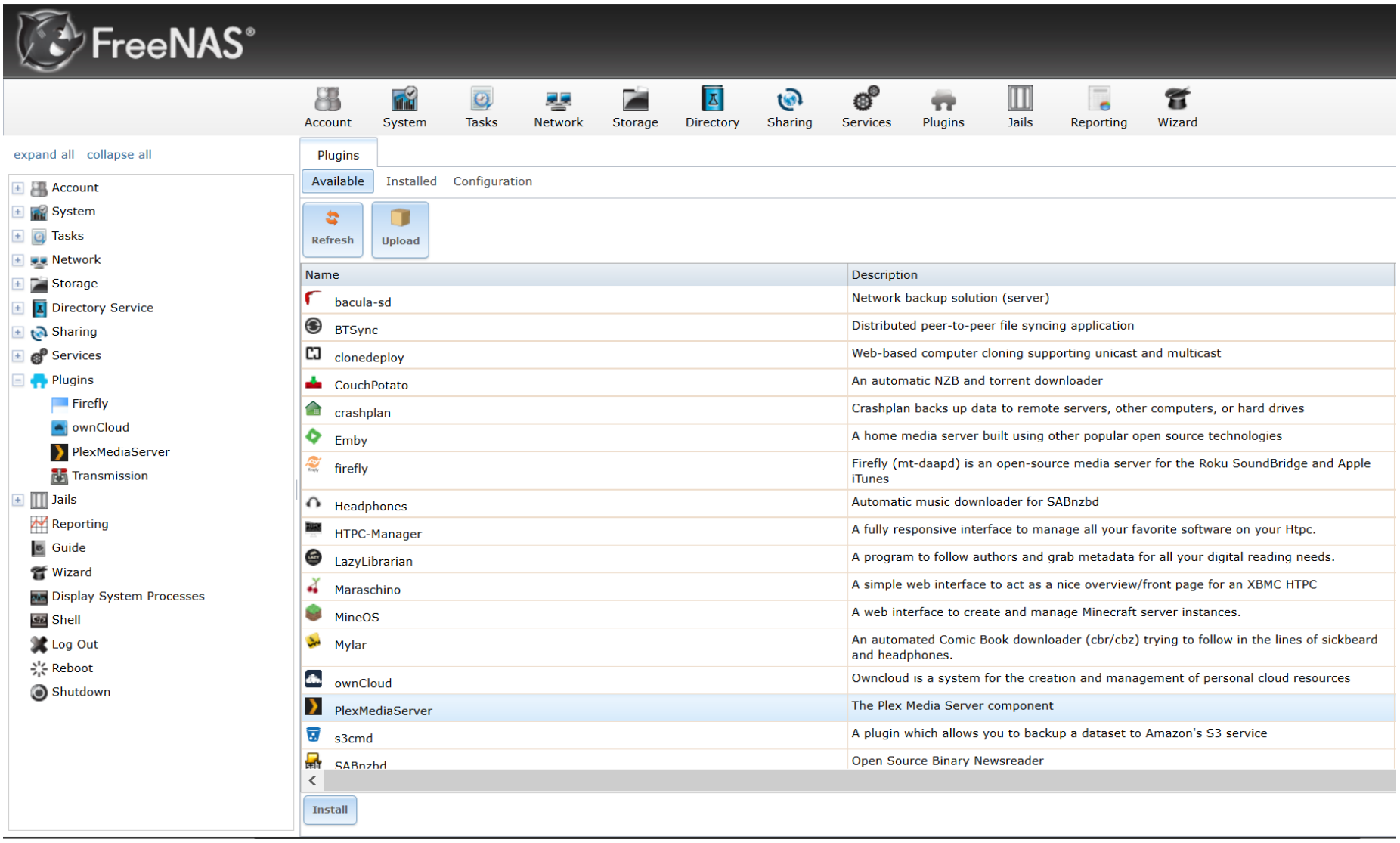


Figure 2 - Installing a Plugin.

Once the installation is complete, the plugin will appear in the “Installed” tab (Figure 3). If any updates are available, the menu will show an “Update” button. Simply click on that button, then “OK” in the pop-up menu to update your plugin to the most recent version. If no update button is available, then you are on the most current version available. You can also turn your installed plugins on or off or delete them from this menu.

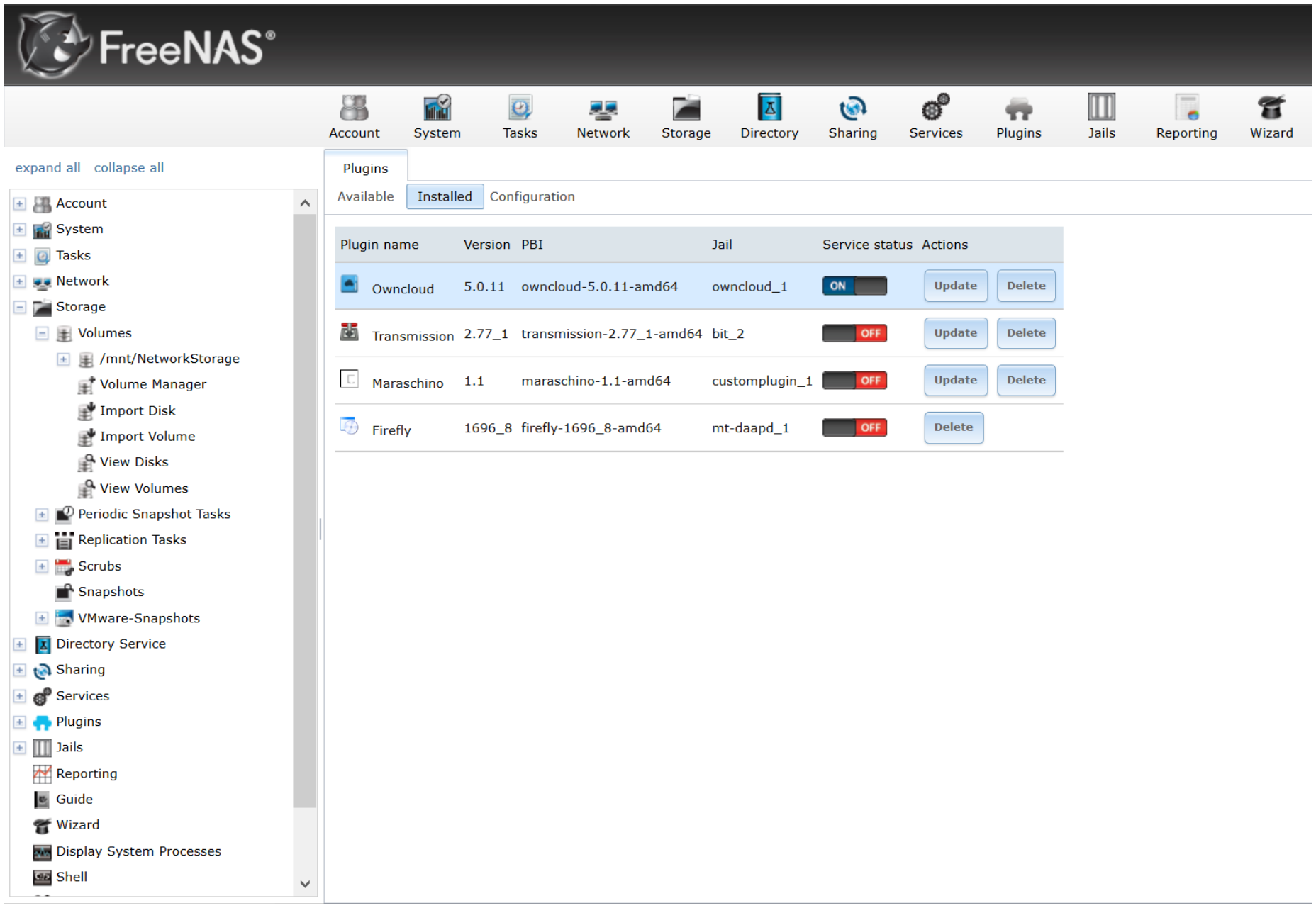


Figure 3 - Installed Plugins Menu.

For full information on the Plugins Menu, go to the FreeNAS Documentation Page at *https://doc.freenas.org/9.3/freenas_plugins.html*.

Plugin Jail Configuration

As of FreeNAS 9.3, when you install a plugin, a FreeNAS Jail is automatically created. This takes out most of the guesswork, but, depending on where you will locally store the files for each plugin, you may want to change the dataset that your jail is linked to. To do this, just go to the Jails portion of the FreeNAS interface, then click on the “Storage” Tab. Once in the Jails Storage menu, select the jail you want to configure and click on the “Edit” button at the bottom left (Figure 4). This will bring up a pop-up menu (Figure 5). Click on the Browse Buttons to edit the Source and Destination folders for your jail and then click the OK button to confirm. You can find full documentation on this process in the FreeNAS documentation at *https://doc.freenas.org/9.3/freenas_jails.html#add-storage*.

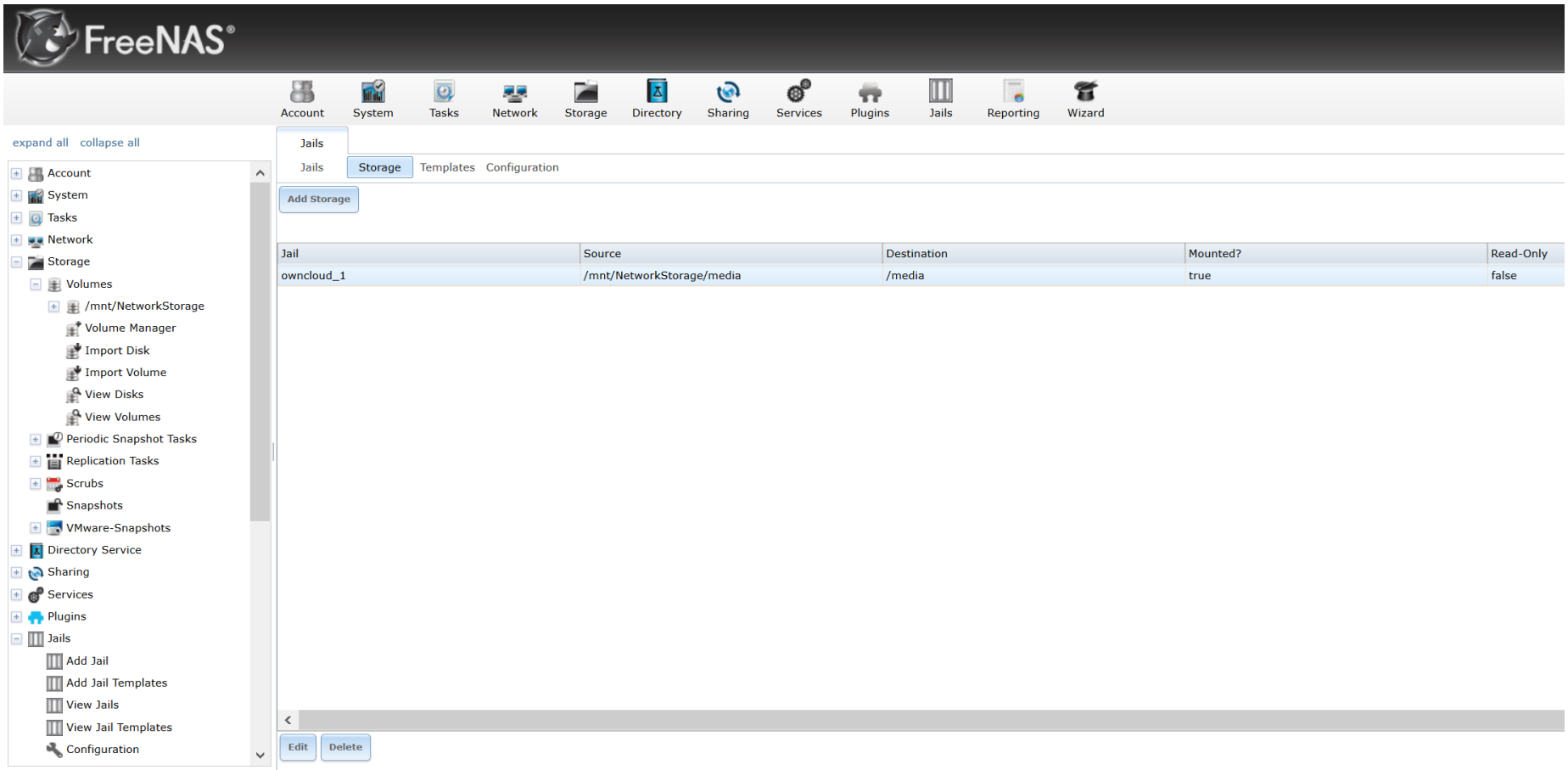


Figure 4 - Jails Storage Configuration.

Figure 5 - Jails Storage Configuration Menu.

Once your plugins are installed and mapped to the desired datasets, you can launch them by clicking on the preferred application in the Plugins section on the left sidebar (Figure 6). This will bring up a pop-up menu with a link to your application configuration (Figure 7).

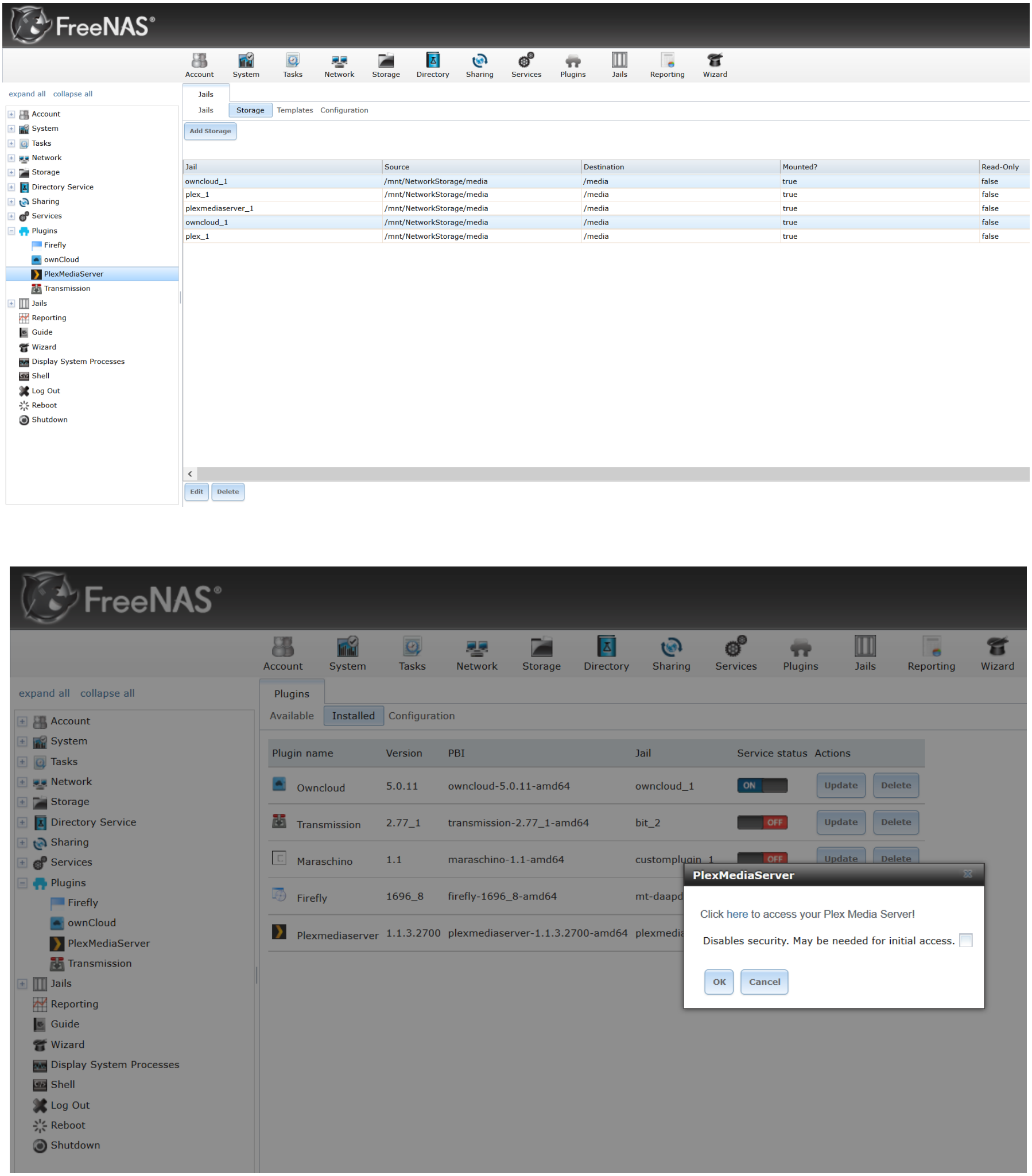


Figure 6 - Launching Your Plugin.

From there, your application will launch in another browser tab. You will need to follow the documentation for each third party plugin for instructions on using the application.

Conclusion

Hopefully, this article will enable you to install and maintain your favorite third party plugins in FreeNAS 9.3.x and 9.10.x versions with ease. This installment concludes the FreeNAS Getting Started series. I hope you have found it beneficial and that it provided you with valuable information for getting your own FreeNAS system up and running in a way that meets all your storage needs.

Additional Resources

Blogs:

- FreeNAS Best Practices: Part 1 | Part 2 | Part 3 | Part 4 - *http://www.freenas.org/blog/*
- FreeNAS: A Worst Practices Guide - *http://www.freenas.org/blog/freenas-worst-practices/*

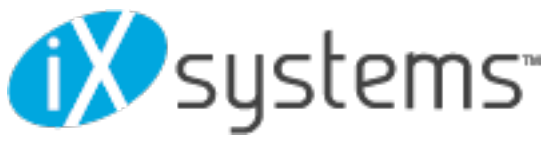
Forums: *https://forums.freenas.org/index.php*

Videos: *https://www.youtube.com/user/FreeNASTeam/videos*

FreeNAS Documentation: *https://doc.freenas.org/*

About the Author:

Mark VonFange has worked for iXsystems since 2008 in various roles including first response for professional services inquiries and developing marketing content. He has been an open source advocate for over a decade & enjoys building and repairing computers in his spare time.



USING FREEBSD AS A FILE SERVER WITH ZFS

In this course, we will learn how to use the current ZFS capabilities to help us build a home file server using FREEBSD 10.3.

Course launching date: 04th of July 2016

What will you learn?

- ZFS administration
- ZFS concepts and features

What skills will you gain?

- ZFS administration basics

What do you need?

- FREEBSD 10.3 with root privileges
- At least 10 GB free space

What should you know before they join?

- Basic FREEBSD administration knowledge

WORKSHOP

Module 1: FREEBSD and ZFS

Introduction to ZFS under FREEBSD

- Why ZFS on FREEBSD?
- ZFS features and concepts

Module 2 title: ZFS Administration

Module 2 description: Cover the commands and features to administrate ZFS volumes

- Create, destroy, list pools
- Zpools: single, mirrored, raid
- Understand ZFS properties

Module 3 title: Putting it all to work: Hosting our files using ZFS

Module 3 description: With the previous acquired knowledge, create a plan on how to organize our files and pools to host our files.

- Set ZFS properties based on the content of the files to host
- ZFS tuning
- Create a File Server using our pools

For more info visit our web page:

https://bsdmag.org/course/using-freebsd-as-a-file-server-with-zfs-2/

Don’t hesitate to ask your questions at

marta.ziemianowicz@bsdmag.org

The most important thing for beginners is to find a community they feel comfortable with.

Alexander Todorov, Founder of Mr. Senko

by Marta Ziemianowicz, Marta Sienicka & Marta Strzelec

[BSD Magazine]: Hello Alexander, how have you been doing? Can you introduce yourself to our readers?

[Alexander Todorov]: Hello, I've been keeping myself busy working on a number of open source projects, otherwise, I've been doing great. My professional background has revolved entirely around open source for the past 15 years including translating software and documentation, community work, development and testing.

My first interaction with open source (in fact it was FreeBSD) was in early 2000 when I was studying the UNIX operating system at university and later became interested in parallel programming and SMP systems. However, I've moved away from that and for the past 10 years my passion has been testing and developing open source software.

[BSD Mag]: Can you tell us something about your company, Mr. Senko? Where the name of the company comes from?

[AT]: Imagine that you are a developer working for a product or service company. Most often you make use of various open source libraries and frameworks to build the products and services in question. Once you hit a bug or missing functionality you have a few choices:

- fix it yourself and contribute back;
- report it and wait for upstream to fix it;
- work around it by any way possible.

Unfortunately, from what I see, many people prefer to work around the problem and forget about it instead of spending the resources to properly investigate, fix and contribute back to the commu-

nity. On the other hand, some communities have not been very responsive, which kills participant's motivation.

Mr. Senko is a start-up that addresses these problems by providing long term support and development for various open source libraries. It's like having your own go-to open source fix-it guy!

[BSD Mag]: You are "Open Source Wizards". What does it mean?

[AT]: Let me first tell you about the name. Mister(ious) Senko was the stage name of a famous Bulgarian magician who lived during the 20th century. I was born in the same city as him so the name came naturally!

Mr. Senko was founded by four friends who've been working a lot with open source. Among us we probably have around 50 years of experience. We call ourselves wizards because we know our technology fields very well and can make bugs disappear. The slogan also plays nicely with the magic theme from the company name :).

We strongly believe our team is our biggest asset and the name also reflects that. It's kind of cool that at the end of the day we can refer to the whole team as if it was a single person with superpowers.

[BSD Mag]: Can you tell us more about your friends that founded Mr. Senko? How did you all meet?

[AT]: The other co-founders are Alexander Kurtakov, Minko Gechev and Krasimir Tsonev.

Alex has been an Eclipse developer for the last 15 years. He is also a member of the Eclipse Platform and Tools Program management committees and serves on the Eclipse Architecture Council.

Krasimir is a senior front-end developer, blogger and speaker who loves writing JavaScript. He is also the author of two books: Node.js Blueprints and Node.js By Example.

Minko is an experienced JavaScript developer and frequent conference speaker as well. He is the author of the popular AngularJS Style Guide and AngularJS in Patterns! Recently he wrote the book Switching to Angular 2.

Krasimir specializes mostly in ReactJS while Minko is part of the AngularJS team.

We've all met through our open source work. Because of this, we've been part of the same communities, visited the same events or worked together on the same projects and that's how we came to know each other.

[BSD Mag]: Which open source systems do you use and why?

[AT]: Well, there are many, really. I personally work mostly in Python and a little bit in Ruby. From the Python world, there is Django and Pelican.

[BSD Mag]: Which one would you recommend for beginners? And why?

[AT]: Well, it depends on the person's intentions of getting involved with open source in the first place. If you'd like to just use some tool and get your job done then I guess use the best tool you can find.

If you'd really like to contribute back to the community, search for a medium sized project that is actively maintained and has several active contributors. These kinds of projects tend to be more responsive and easier to join for beginners. There are enough experienced contributors from whom you can learn but there is also enough work to be done so that your contributions are recognized. You can also try finding a mentor from a project you like and go from there.

I'm not going to recommend any project in particular because it all comes down to personal preference. The most important thing for beginners is to find a community they feel comfortable with so they can focus on technical work.

[BSD Mag]: Do you have any favorite one?

[AT]: Somebody once told me that if I've taken a few weeks to build a website in Django I'm experienced enough to do anything with it. If I've been using it for 3 months then I was already an expert. I think this is mostly true and Django is one of the frameworks that is always on top of my list.

I also have a least favorite one, it is Ruby's RSpec test library. What I don't like in particular is the fact that the same thing can be expressed in multiple ways and when it comes to describing test mocks you can do this very subtly in just a single line. The problem arises when you have to work with inexperienced developers who get confused about this freedom of choice or don't really understand the hidden mechanics behind that one line of code I mentioned earlier. In these scenarios, I'd prefer to have less freedom (only one way of doing things) or be more explicit, more verbose so it is obvious what is going on under the hood.

[BSD Mag]: What do you think about the open source community? Have you ever contributed to one of them?

[AT]: I think the open source community is the greatest thing that has happened in the software industry for the past 20-30 years. All of the innovation and technology that we see today is the result of many communities. I also strongly believe there is lots more to see from the open source model applied outside the software industry.

I'm a heavy contributor to open source. Recently, I also became the maintainer of django-chartit and effectively revived the project by merging stale pull requests and personal forks. In general, whenever I see an issue I try to patch it and send a pull request.

Another project that I'm currently heavily involved with is called Cosmic Ray. This is a mutation testing tool for Python. Being a somewhat research topic, mutation testing is still not heavily used in production so the tools lack some practical features. I just happened to have a need for these features so I've made quite a few pull requests and will continue to do so.

[BSD Mag]: What is QA and what does it mean that you are a QA Guru?

[AT]: QA stands for Quality Assurance or sometimes Quality Assistance. This is the science of testing software. My job is to make software better by exploring and finding bugs, writing test cases and improving the code. I guess it is more accurate to say I'm a software developer with more focus on quality than features. QA Guru is just a vanity title really. It's just a way of saying I've been around bugs for a long time and I find it relatively easy to spot problems in software.

[BSD Mag]: Any particularly gruesome bug stories you could tell us?

[AT]: One example of a high impact packaging bug is Django #19858. During an urgent security release, it was discovered that the source package was shipping byte-compiled *.pyc files made with a newer version of Python (2.7). Even worse, there were byte-compiled files without the corresponding source files.

Being a security release, people rushed to upgrade immediately. Everyone who had Python 2.6 saw their website produce ImportError: Bad magic number and crash immediately after the upgrade!

Another one is s3cmd #668 that was caused by backward incompatible change in Python libs. I just like how subtle this is. Python introduced two new parameters to a class constructor, with default values and all. However, they also slightly changed the behavior based on those new parameters and essentially killed backwards compatibility.

These are some examples of what I call „Hello World Bugs“ - bugs that are not related to the complexity of the software under test but have to do with the external environment in which the software operates.

[BSD Mag]: What are the challenges your company has to face at the moment? Any plans for the future?

[AT]: Right now, Mr. Senko is working with a few pilot customers. The challenge at the moment is finding more suitable pilot customers so we can steadily get off the ground. Our plans are to have at least one technology stack fully supported and operational by the end of the year. Future plans include growing the company and adding more technology stacks. We're also looking at some non-orthodox technologies, like .NET.

[BSD Mag]: Any piece of advice for our readers?

[AT]: Thank you for your contributions to all readers who currently work in the open source field and keep up the good work! If you happen to be wondering if working with bunch of random strangers online is your thing my advice is to give it a try!

Happy coding and I hope to meet you some day!

About Alex:

Alex has been testing open source software for the past 10 years and is responsible for finding more than 1,600 bugs! He is also a general purpose open source developer, blogger, speaker and an entrepreneur! Alex is sharing his knowledge by training and mentoring students from Sofia University at local workshops and hackathons. You can find him as **@atodorov_ on Twitter**.

Strengthen and consolidate what you are good at.

Franck Porcher, Founder of Francky's Computer Engineering

by Marta Ziemianowicz, Marta Sienicka & Marta Strzelec

[BSD Mag]: Hello Franck, how have you been doing? Can you introduce yourself to our readers?

[Franck Porcher]: Hi, I'm Franck, 55. I grew up in Paris, France, where I got my formal academic education and started my career as a research engineer in computer science (CS) for about ten years until my early thirties. Then I left Europe altogether to roam the world with no plan to return, having since found a nice balance of life between the Pacific mountains of British Columbia, Canada, and the pristine South Pacific Polynesian Islands of Tahiti where I settled permanently in early 95, writing software and teaching CS bachelor courses in l'Université Française du Pacifique for a living, never missing a moment to enjoy the Polynesian way of life, its exuberant virgin nature and amazing lagoons :)

[BSD Mag]: It looks like you have been involved in many interesting projects! Let's begin with Francky's Software Engineering. What is the company about?

[FP]: «Francky's Computer Engineering», Francky's for short, is a start-up specializing in open source solutions that I established in 2011 to transpose and develop in Central America the same activities that I had been carrying-out in the start-ups I created and was operating in Tahiti.

In the true spirit of its founder, Francky's is a technology-driven company dedicated to produce software of the highest quality, committed to state-of-the-art coding, agility and best practices, bound to cutting-edge knowledge and creative problem solving, always overcoming conventional thinking and methodologies to convey elegance in programming as a form of artistic expression. Our mission in one short sentence: design & deliver uncompromised IT services and solutions :)

[BSD Mag]: What have you been doing in Tahiti?

[FP]: I settled in Tahiti in 1995 well before the rise of Internet. Noticing that the place was lacking reliable high-tech IT services, I jumped at the opportunity and created my own start-ups, offering the community a broad range of high-quality integrated IT services, including infrastructure solutions, Web and custom software development, as well as professional training.

Through commitment and hard work, we met an immediate and long lasting success, even after competition started to settle in from far-away France. At that time, our professional reputation was indeed firmly established, in particular as the pioneer of open source solutions in Tahiti, and our loyal customers – virtually all the largest companies in Tahiti – always retained our services, thus preventing us from ever suffering from the competition.

To date, though the 2007 sub-prime crisis significantly slowed down the local economy, opening the market to lawless mercenaries, we are still vividly remembered as the guys who introduced open source and modern IT in Tahiti, constantly pouring in innovative solutions, for instance, full-blown e-banking platforms for two local banks out of three (Web+Wap+SMS services, CRM & VoIP, starting using GNU Bayonne before the birth of Asterisk), and Linux Small Business Server, a full-blown custom Linux Server distro providing a web-administered, integrated Internet infrastructure solution, deployed in the city councils of most municipalities of French Polynesia, and still in use today, 15 years later.

[BSD Mag]: What about people in Tahiti? Did you find many promising cyber security/IT specialists?

[FP]: Tahiti, the heart of French Polynesia, has long been one of France's most strategic colonial trading posts, given its vast maritime domain – that raised France as today's second international maritime power – and its remoteness that led France to “safely” use it as a testing site to develop its (military) nuclear program. Unfortunately, France's ambition was never to develop Tahiti's infrastructure so it could emerge as one great intelligentsia. As seems to be customary of colonized societies, the general acculturation and its dramatic social consequences are no exception in Tahiti, far from it.

IT mostly appeared in Tahiti as a new commercial service, not as an art or a science, or very late and very succinctly. For these reasons, we can not possibly talk of Tahiti as a whole as a place to expect finding real, genuine and long-lasting expertise in security or cyber-crime. Finding accomplished IT engineers in Tahiti has always proven an almost impossible task for our start-ups.

[BSD Mag] What about L'Académie Informatique?

[FP]: L'Académie Informatique is my latest start-up, registered in Tahiti to promote, develop and provide high-tech professional open source IT training in Tahiti, an essential service that remains

mostly lacking locally. However, because of my commitment to other projects, including frequent ins and outs, this new activity is not growing as fast as I would like.

[BSD Mag]: Apart from lack of time, did you encounter any other challenges with this project?

[FP]: Knowing which topics to train professionals with could be a challenge for some, given the large inventory of available IT tools, techniques and technologies.

However, in our particular situation, we have built the confidence it takes, based on decades of knowledge, strong practice and wide experience, to know that the topics we are selecting have long-proven to be premium choices that will remain modern, useful and up-to-date, up-to-the-job options for the longest time, options we feel are very sound for any professional to invest time, money and energy in learning them.

[BSD Mag]: I'm always very impressed, if somebody has a patent. What have you patented?

[FP]: I co-own two patents, nothing extraordinary, in the field of vision and neural computing back in the early 90's when I was a researcher and lead developer at Dassault Electronique, sister company of France's topmost flagship aviation industry. One of them is related to an automatic classification system of vehicles across tolls (public patent), the other to a signal processing system for guiding self-steering flying gears (classified patent).

I also benefit from an industrial protection from France's National Intellectual Property Institute, related to an integrated messaging system solution that revolves around the concept of unified communication channels (Email, SMS, Fax, VoIP): SMSeXpress (2002).

[BSD Mag]: Your first degrees are in mathematics and physics. What turned you in the IT direction?

[FP]: Mainly chance I would say!

As a full-time resident student of mathematics and fundamental physics at the University of Orsay, near Paris, France, not quite concerned by the hype of social life, I felt myself having plenty of time to enroll in as many courses as I could beyond the mandatory courses. One of them sounded magical: "Numerical algorithms and Programming". Though places were limited, I was luckily accepted. Please remember that I'm talking about France back in 1979, where computers were mainly mainframes and unheard of. I was, of course, no exception, having never come across any, only fantasizing about what they were…

The course focused on programming numerical algorithms using Fortran, something which I found very recreational. Orsay campus' computing center, an impressive building, revolved around a massive Univac 1100/80 mainframe computer system and its huge bay of peripherals, where we would enter the programs using punch cards. I got hooked, turning all my free time into programming and learning the intricacies of the Univac 's operating system from the thin English technical documentation available! I may have done well enough since I also managed to secure a student job there for two years as a console operator doing mostly night and week-end shifts, something fabulous that gave me lots of unsupervised time to freely access all the computing resources at no cost :)

This early experience with computers, however, changed my path forever, given I found the fun I was experimenting in programming computers order of magnitudes greater than scribbling kilometers of dry maths and physics on pads of paper. Though I pursued my initial curriculum so to graduate in mathematics and fundamental physics, I quickly branched towards computer science and eventually completed a Ph.D on designing and implementing a new scheme of languages based on logic and symbolic constraints for knowledge representation.

[BSD Mag]: Do you have a favorite programming language? Or one that is particularly difficult to learn?

[FP]: I have mostly four favorite programming languages that I have used almost daily for decades.

Though I was first introduced to Fortran, Pascal and C – nothing short of fine languages – my first great true love became Lisp, which has always bent to conform perfectly to my warped way of seeing things and solving problems. I used Lisp extensively in the 80's, including my own F-Lisp system, from fast-prototyping to developing language processing tools. I still use CLOS and Scheme today, therefore, feeling very supportive of the Clojure initiative, though I don't have a formed opinion whether targeting the JVM was/is a good thing or not.

My second great true love is Perl 5, discovered 22 years ago and extensively used daily since, in any sort of unimaginable ways and complex software projects. I believe Perl 5 offers today's greatest regex engine and Unicode support. I love Perl 5 for the bunch lot it gets more than right compared to other (mainstream) languages. I like Perl people and the Perl culture, its excellent textbooks – try “Computer Science and Perl Programming” or “Higher-Order Perl” for some flavor – and its wonderful CPAN repository. It may drive me nuts sometimes, then I love it more and more for teaching me “better” and “smarter” ways. Did I say I love Perl 5?

Last but not least, Erlang, my third true love, an amazing mature language and framework, very modern, which, like Haydn or Clementi's piano sonatas vastly ignored by most mainstream pianists for no possible artistic reason but mere audience ratings, is vastly ignored by most develop-

ers for no good reason but laziness and lack of good taste.

And of course a special mention to Bash, an amazing language for system programming and administration, which, together with Perl, makes an outstanding FreeBSD developer's Swiss knife!

As for languages particularly difficult to learn, I believe they are all those that lack focus, vision, as well as some sort of elegance and orthogonality. In that respect, please forgive my boldness, Basic (early 80's) and PHP may be the most offending representatives...

[BSD Mag]: What about open source? How has it started and from which system?

[FP]: I starting my professional career in the mid 80's as a software engineer for Dassault Electronique. Though the company always provided outsized IT computing capabilities (at the time, no computer resources, either hardware or software, ever came cheap), I always felt limited by the inventory of software I could use, leading me at times to develop my own tools on the side (e.g. the modern F-Lisp system).

As a result, I died and went to heaven the first time I got my hands on a full-blown Slackware Linux 1.0.13 distro years later in early 1994, which came as a large set of 1.44MB floppy disks! Though personal computers were still very slow (Pentium 60 Mhz, 64 MB RAM, HDD 256 MB), a properly tuned Linux system would shine compared to any mainstream Windows 3.1/3.11 system.

Since that day, I have exclusively, and most happily, resorted to using open-source software, making only a subsequent switch from Linux to FreeBSD in 2006. These fabulous systems have never let me down. Over time, they have given me the freedom to experiment and to create whatever piece of software crossed my mind, allowing me to promote open source solutions to further help people and local communities. Over the last two decades, often dubbed a workaholic maniac, I have worked on hundreds of projects and written well over a million of lines of code in many languages.

[BSD Mag]: What made you switch to FreeBSD in 2006?

[FP]: At that time, the insanely growing number of Linux distros and the underlying inconsistencies/incompatibilities between one another (package management is one of them) was starting to get out of hand and to become an intractable proposition; something still quite true today, one had come to never quite know what to expect upon encountering and using a random Linux box!

This was also the time I started to spend more and more time upgrading my Linux boxes, fighting corrupt updates and broken packages – I won't reveal the name of the distros I was using at that time, but let me say they all were mainstream... ;). Though I had worked with Linux for over 12 years, I decided that I had it with it and that the time had come to find a more consistent, more sta-

ble and more reliable alternative. FreeBSD gave me exactly the break I was waiting for!

Though it took me some time to acquaint myself with FreeBSD as a whole and to start to appreciate and rely on some of its lovely subtleties (e.g. the existence of a true central repository – a single place where you can find the entire operating system sources, including all older versions – its OS/World dichotomy, as opposed to kernel/everything-else under Linux, its ports and dual way to manage it), I have quickly come to trust FreeBSD to always provide me with (almost) the ultimate in stability and security, higher performance, as well as technical documentation of great quality and usefulness. In the end, I found developing with FreeBSD much more enjoyable, enlightening and professional.

[BSD Mag]: Do you have any open source software that you like the most?

[FP]: My average daily work involves using a vast collection of free software. Amongst them:

FreeBSD, development tools (Bash, LLVM toolchain, Screen, Terminal, Vim –I love scripting Vim), infrastructure (Asterisk, FreeSwitch, OpenSSH, Qmail, Tor), languages (Bash, C, CLOS, Clojure, Erlang, Perl, Scheme, JavaScript and its ecosystem), databases (PostgreSWL, MongoDB), end-user applications (Firefox, Thunderbird, The Gimp, Jack and its ecosystem, LibreOffice suite, VLC).

Amongst these tools, I don't think I could manage today without Bash, C, Erlang, FreeBSD, LLVM, OpenSSH, Perl and Vim (and the multiple-tabs terminal of course)

[BSD Mag]: What about the community? Have you ever been a part of it?

[FP]: Contributing back to the open source community has always be in my mind. However, as small-scale IT engineering companies dedicated to produce great solutions, never benefiting from public grants or funding, yet organizing conferences and workshops (Richard Stallman was our guest speaker in Tahiti for the “Linux World Days 2000”), I could never make it a priority (releasing reliable material always takes a lot of time). However, I have published and talked a few times in various OSDC and YAPC conferences, and I am now working on cleaning-up stuff I can give back to the community, including some Perl modules to be released on CPAN.

[BSD Mag]: Is there is any challenge your company is facing at the moment?

[FP]: Not really, at least nothing really new:

1. Remaining uncompromised and independent.

2. Keeping pace with the rapid evolution of technologies, tools and trends.

3. Cost of lobbying companies to switch to open source solutions.

[BSD Mag]: Any plans for the future?

[FP]: Launch “TIMIT”, the International Minoan IT Bootcamp, an educational summer-time initiative set in warm and exotic Crete, aimed at tutoring promising technologies through expert workshops and group IT projects.

[BSD Mag]: That project sounds really interesting! Can you tell us more about it - who can enroll in this Bootcamp, what expectation do you have for this project?

[FP]: The project is still at its very infancy, and much remains to be done before we launch it.

However, this project means a lot to me; it is truly about our long-term dedication to transfer knowledge and skills, especially in the field of computer languages, to pass them on and help others harness the essence and quintessence of these tools in the hope to produce innovation and reveal new vocations.

We envision these boot-camps as “master-classes” (often used in the context of performing-arts, especially music) aimed at gathering together small groups of people driven by the same passion and the same desire to learn and improve, in the form of dedicated and committed small-scale workshops (e.g. 20 participants for a whole week) held in exotic settings (imagine resort in warm, exotic and beautiful Crete) as a way to boost endorphin rushes -- that is excitement, joy, happiness, well-being – and stimulate deep learning and creativity.

We are planing these boot-camps to be as accessible and open to anyone as possible. However, enrolling would probably require some kind of personal background experience so to maximize the benefits of the provided inputs.

[BSD Mag]: Do you have any piece of advice for our readers?

[FP]: The most valuable lesson that I learned very early on is that at the end of the road “Quality always comes cheaper than non-quality”.

In software development, a complex but exciting job, quality often means better designs, better use of appropriate technologies, more innovation, more reliable solutions, better time to market, less maintenance and downtime, happier clients and customers... a better bang for the buck!

For instance, a one-liner can be cheap, but it does not have to be dirty!

Doing quality is not technically complex, and certainly no more complex than doing dirty work. What is complex in doing quality is to decide it and develop the right attitude to support it: personal will, knowledge, commitment and vigilance, and the love for the work.

On the realm of knowledge:

1. Figure out the topics you are best at.

Interview

Do not waste energy trying to improve topics you are weak at; strengthen and consolidate what you are good at instead. Tip: should a weak topic become important to your practice, learn and improve it, otherwise forget it altogether!

2. Keep expanding your knowledge on these topics through experience, tools, best practices, use cases... Be curious and read as much as you can, that is mags, text-books, forums, etc. Tip: try to never end a day without the feeling you have learned something new and valuable.

Thank you.

About Franck:

Franck started his career as a researcher then reoriented himself as a start-ups entrepreneur, university professor and general purpose open source developer for the last two decades.

You can easily find Franck as *https://bz.linkedin.com/in/franckporcher* on LinkedIn.

Email: *franck.porcher@gmail.com*

Rob's COLUMN

The UK Sunday Times reports that the Reuben Brothers are in advanced talks to sell a 50% stake of Global Switch in a £5bn deal to an Asian consortium, with a potential option to sell the remainder at a later date. With the increased interest China is showing in Western infrastructure, is the cloud becoming an even greater security risk to national security?

by Rob Somerville

Those who closely follow geopolitics and global business will not be surprised by the ongoing shift in the balance of power towards the East, both from an economic and technological perspective. While originally conceived in the 1930's, with the adoption of Neoliberalism by Thatcher and Reagan, the aggressive onward march of de-regularisation, privatisation, outsourcing, and all the other attendant evils associated with the "free market economy", has now jumped the gate from a national to a global phenomena. Few realise the pernicious dangers associated with the crushing of the Keynesian and national economic models, nor do many appreciate that the society where Neoliberalism was extensively tested was Chile. Alarms bells should ring when one considers a quote from one of the fathers of the concept - Friedrich Hayek – that "my personal preference leans toward a liberal dictatorship rather than toward a democratic government devoid of liberalism". Twinned with horrors of the Pinochet regime, the only conclusion a rational being can come to is that both models (dictatorship and democracy) are indeed deeply flawed and choosing between them is a case of the lesser of two evils. While much is said about the so called Chilean economic miracle, little is said about the shift in power between the poor and the rich, with 44.4% Chileans living in poverty by 1990. And despite all the hyperbole, there are those that have a quite few issues with Imperialism and Western democracy as well.

Economics, politics and social justice aside, the biggest danger of Neoliberalism is that once adopted, a system cannot easily go back to a central or state run model without serious trauma. What was once owned by government and paid for by previous generations is transferred to some unaccountable corporation or hedge fund. Or to put it more crudely, privatise the profits then socialise the losses. Nowhere is this tension more prevalent than in the technology industry, and specifically from a national security angle, critical infrastructure. And the cloud, by the hour, is morphing more and more into critical infrastructure. While the bureaucrats and lawyers, etc., mutter about data sharing agreements, walled gardens, data protection

and international law, Wall Street and the global financiers are having a feeding frenzy at the mergers and acquisitions trough. Having dealt with a multitude of IT suppliers over the years, I doubt if I can count on two hands where a vendor has gone through this process and the customer has benefited in improved quality of service. A classic example of this, of course, is HP and Compaq, companies I would have no hesitation, in the past, lauding them for their quality, service, and innovative products. Yes, they both suffered from internal problems that forced them to change their business models. Maybe I am just too sentimental, but I miss the age where the focus was on engineering excellence rather than service provision and profit for the market's sake. This shift raises serious concerns for the customer, as more than ever, you cannot accurately predict the direction where a vendor will end up in 5 or 10 years time. While this has always been so in the technology sector, the pace has accelerated, with all the attendant risks and uncertainty. And with few exceptions, rising costs, as in the consumer markets, as well.

Much has been lauded about Open Source software, yet very little attention has been paid to the hardware and infrastructure side of the equation. On a crude level, hardware is just a physical collection of logic gates and maybe an interface or two, yet few apart from the manufacturer or designer understands what is going on underneath. As an electronic hobbyist in the 1970's, there were two types of transistor commonly available, 0C71 and the 0CP71. The latter was considerably more expensive than the former, as the latter was photoelectric, and its electrical characteristics were affected by light. It was well known in electronic circles that the former could be transformed into the latter by scraping away the paint with a scalpel. With hindsight, this is the latter day equivalent of software hacking. If the IP mandarins were around in force then, I probably would have been carted away. The key point here, though, is simple, you need to guard your vertical and

specialist engineering knowledge or else the law of unintended consequences takes affect. Why else would the US have such strict controls over the export of strong encryption in the 1980's?

Regardless of where we look, be it technology, finance, power provision, water supply, media or health, the Neoliberalism agenda is replacing the old order of control with international monopolies, and there is very little we can do to close the door. Appeals to the politicians are no more than a small voice in the wilderness, as the weight of the corporate lobbyist bears down on the democratic process. It matters not what nation is the instigator here - Chinese, Russian, French, British or whatever - the goal seems to be to disrupt any cohesive accountability and original vision, then enforce and shift the power base from the engineer to the profiteer. The birth of the middle class came from the industrial revolution and positive social change, often by the inherent spirit of the entrepreneur philanthropist, who, often for selfish reasons, realised that an educated, healthy, happy workforce were much more productive than the alternative. This in turn increased the demand for democratic accountability, and the growth of government and the body politic has been one of the success sto-

ries of the 20th century. Sadly, though, the age also brought two world wars, countless International skirmishes and an underlying global instability that has steadfastly refused to go away.

In the UK at least, there is growing unease as people realise the longer term implications of Neoliberalism. The recent pro-Brexit vote has truly put the cat amongst the pigeons, as the politicians, bureaucrats, lawyers and management consultants scrabble to put the Federalist vision back together again. The astute will have observed that the referendum result was not so much an anti-Europe vote, but a desperate cry from a nation emerging from the corpse of a dead empire that no longer has an industrial base, nor a nationalised railway, communications, power or water infrastructure. While the exercise has indeed increased choice, it has also increased cost, and the irony has not been missed by many regarding Southern Rail, who had the audacity to announce £100m profits despite a £20m subsidy from the UK government. Partly French owned, this will no doubt add to the clamour for re-nationalisation.

While these observations may seem rather specious while debating ownership of cloud infrastructure, we need to take a step back and examine the long term implications, especially as this technology is being fervently adopted across business sectors. The recent revelations concerning Volkswagen car emissions, and a number of security flaws discovered by Dr. Sergei P. Skorobogatov in MIL-SPEC devices alone should raise flags of concern. The intent, criminal or accidental, matters not. For any organisation, data is the most important and valuable commodity next to employees. Sadly, where due diligence is a matter of daily life in the mergers and acquisitions market, there is little corresponding rigour throughout the entire technology supply chain, from discussion about motive, right down to physical security and human rights. Even less so in small or medium sized business. We have International certifications and engineering standards, but as proven time and again by forged worthiness certificates, blackmail, corruption and CE markings, there will always be some that place profit and exploitation before all else.

Whilst the cloud brings many benefits, until we learn to make peace with each other, war will continue either through the hands of the soldier, or in the case of technology, via hackers, criminals or hostile foreign powers. Irrespective who we sell the family silver to, it does not take the mind of Sun Tzu or Machiavelli to realise any failure is always harder to manage and control when it is out of your control. And there are always those that are more than willing to set up an innocent third party for their own nefarious ends. Some business sectors really do not export well. The UK prime minister, Theresa May, has ordered the security services to investigate the proposed Hinckley Point UK nuclear deal with China. Global Switch have denied there are any national security issues, but rumour has it they may be placed under the microscope also. Which is just as well, really, as there are "safeguards" in place regarding the ongoing 10 year partnership between the main UK telco British Telecom and Huawei. Of course, the fact that the UK government scrapped video

conferencing kit by the manufacturer due to security concerns is of no relevance. Meanwhile, Huawei is looking at building $10bn worth of cloud infrastructure. Deja vu, anyone?